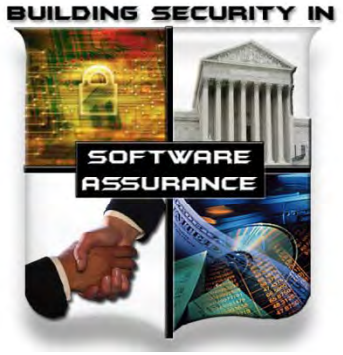
Homeland Security

Commerce

National Defense



Software Security Knowledge: CWE

Knowing what could make software vulnerable to attack

Robert A. Martin
Sean Barnum

May 2011

CWE™

MITRE

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
MAY 2011		**00-00-2011 to 00-00-2011**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Software Security Knowledge: CWE. Knowing what could make software vulnerable to attack	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Mitre Corp,202 Burlington Road,Bedford,MA,01730-1420**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
Presented at the 23rd Systems and Software Technology Conference (SSTC), 16-19 May 2011, Salt Lake City, UT. Sponsored in part by the USAF. U.S. Government or Federal Rights License

14. ABSTRACT

15. SUBJECT TERMS

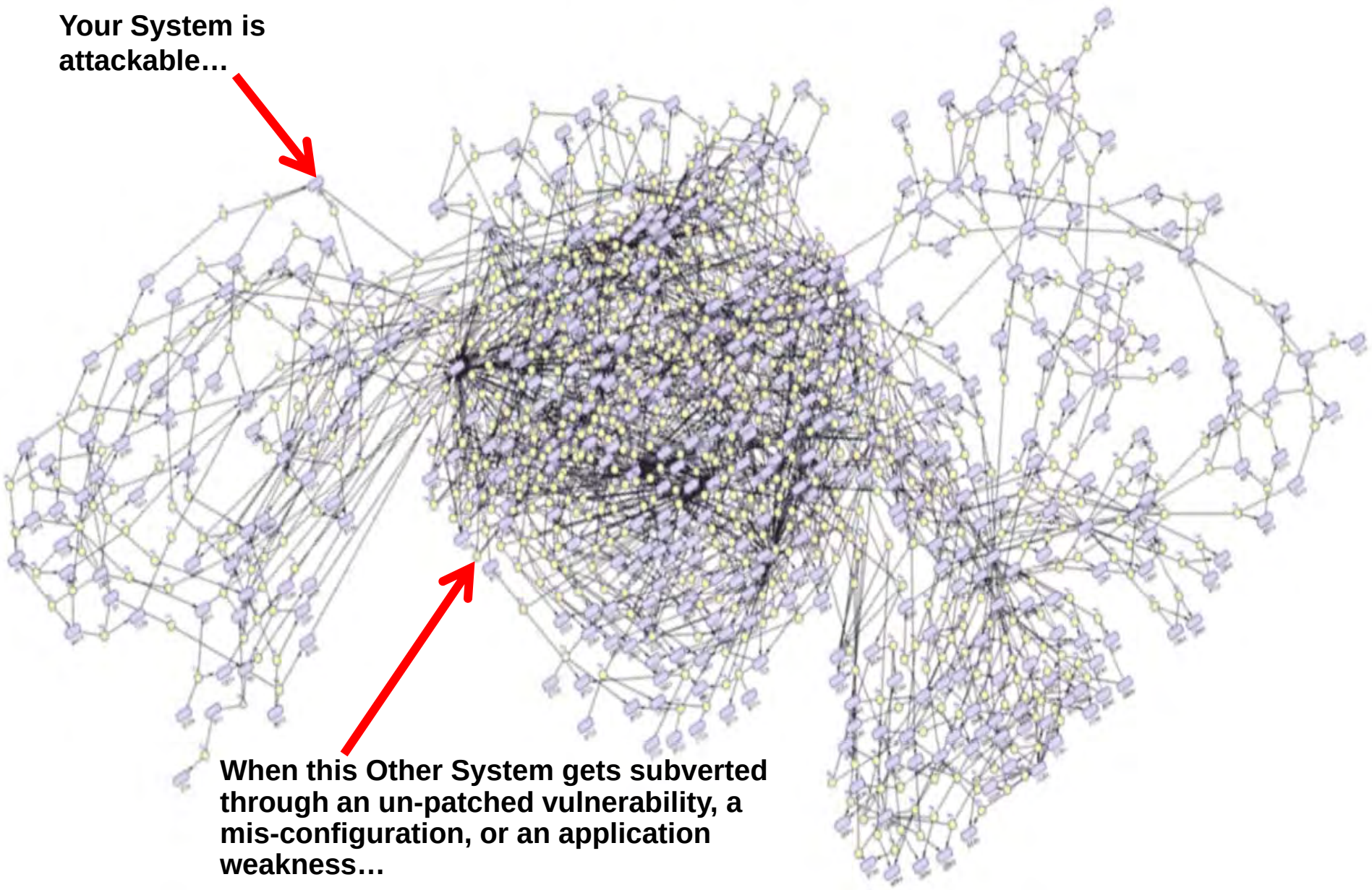
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**60**	

Agenda

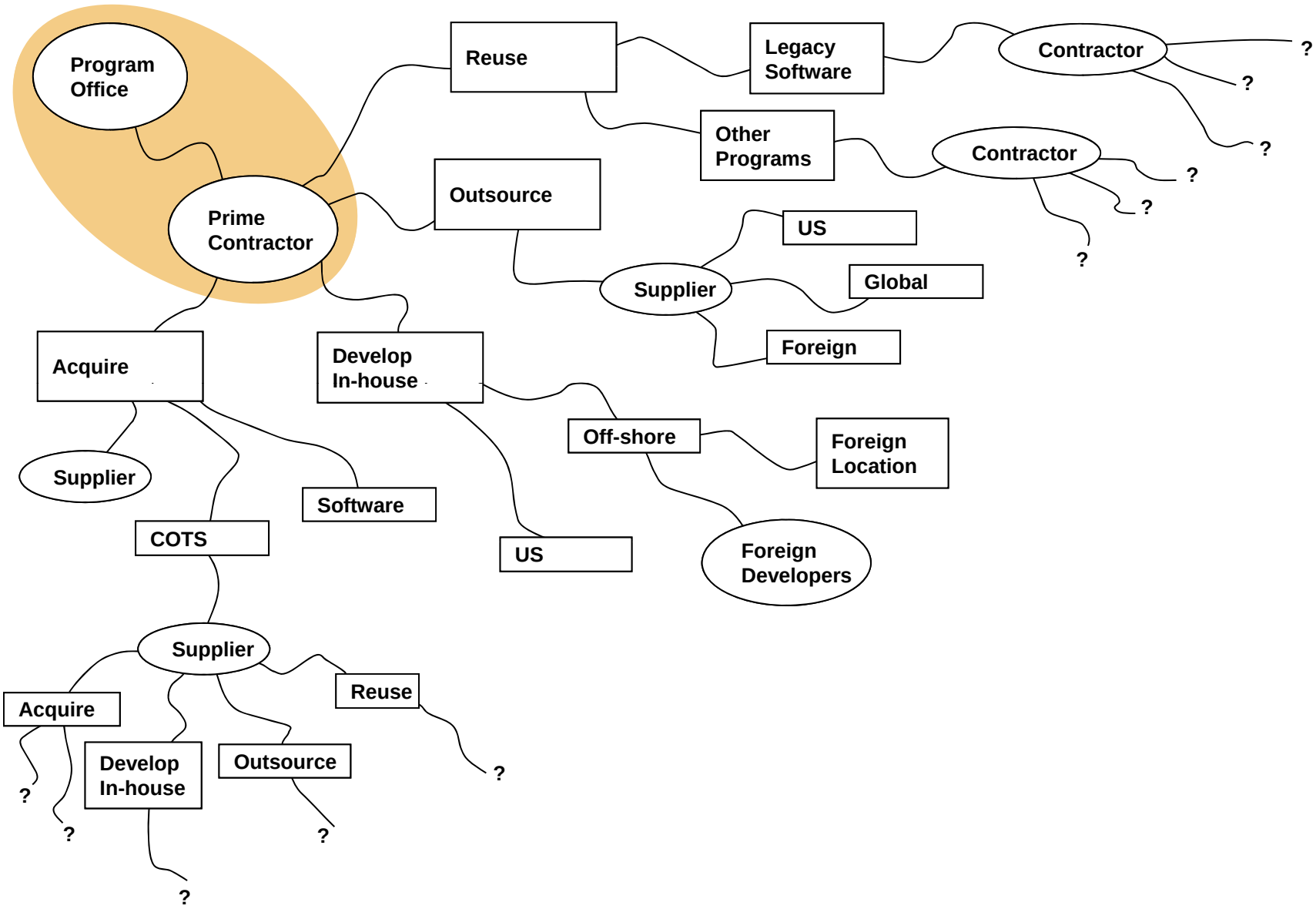
8:00-8:45am	Software Security Knowledge about Applications Weaknesses
9:00-9:45am	Software Security Knowledge about Attack Patterns Against Applications
	Training in Software Security
10:15-11:00am	Software Security Practice
11:15-12:00am	Supporting Capabilities
	Assurance Cases
	Secure Development & Secure Operations

Making Security Measurable™

Today Everything's Connected

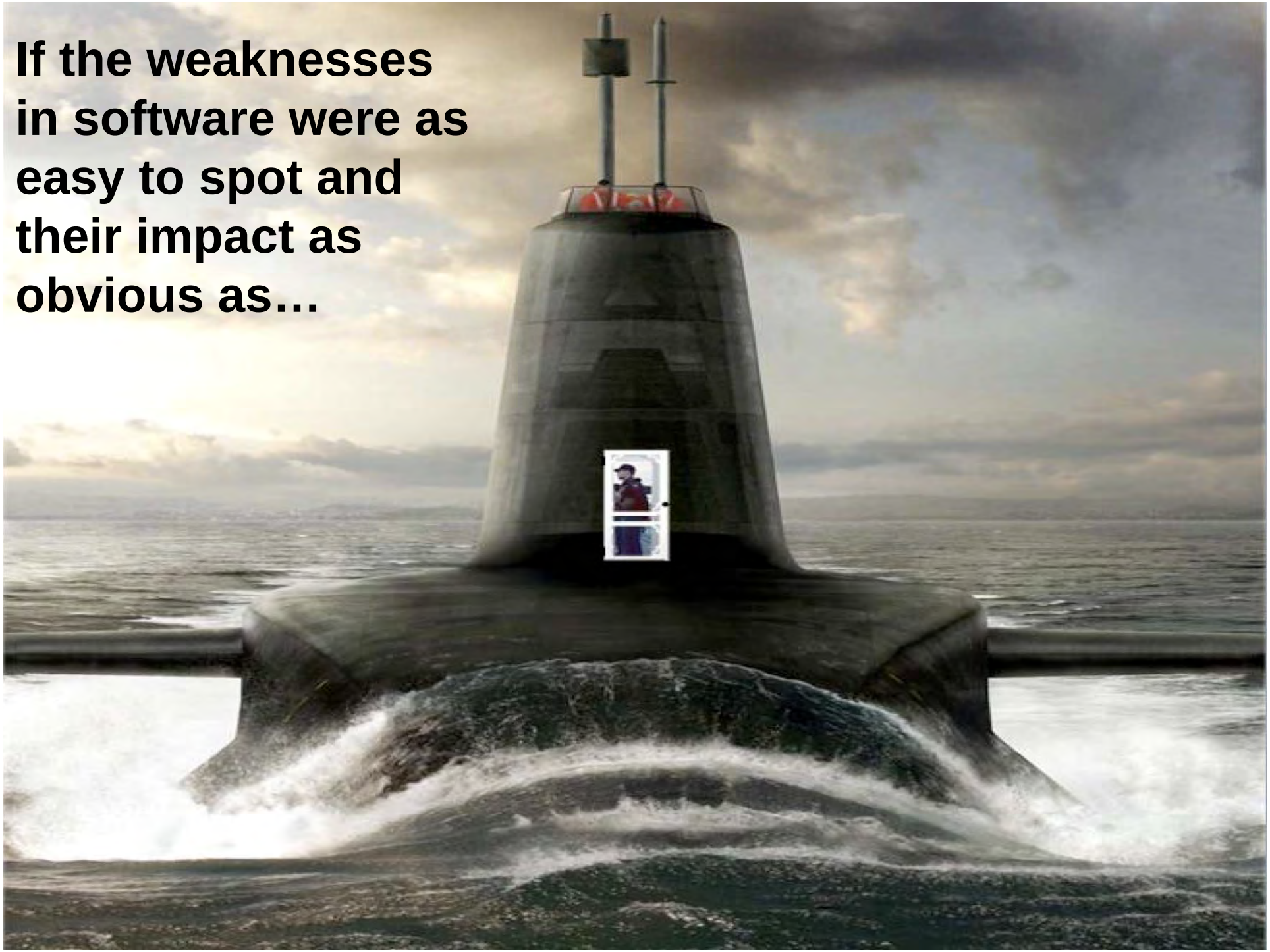


The Software Supply Chain

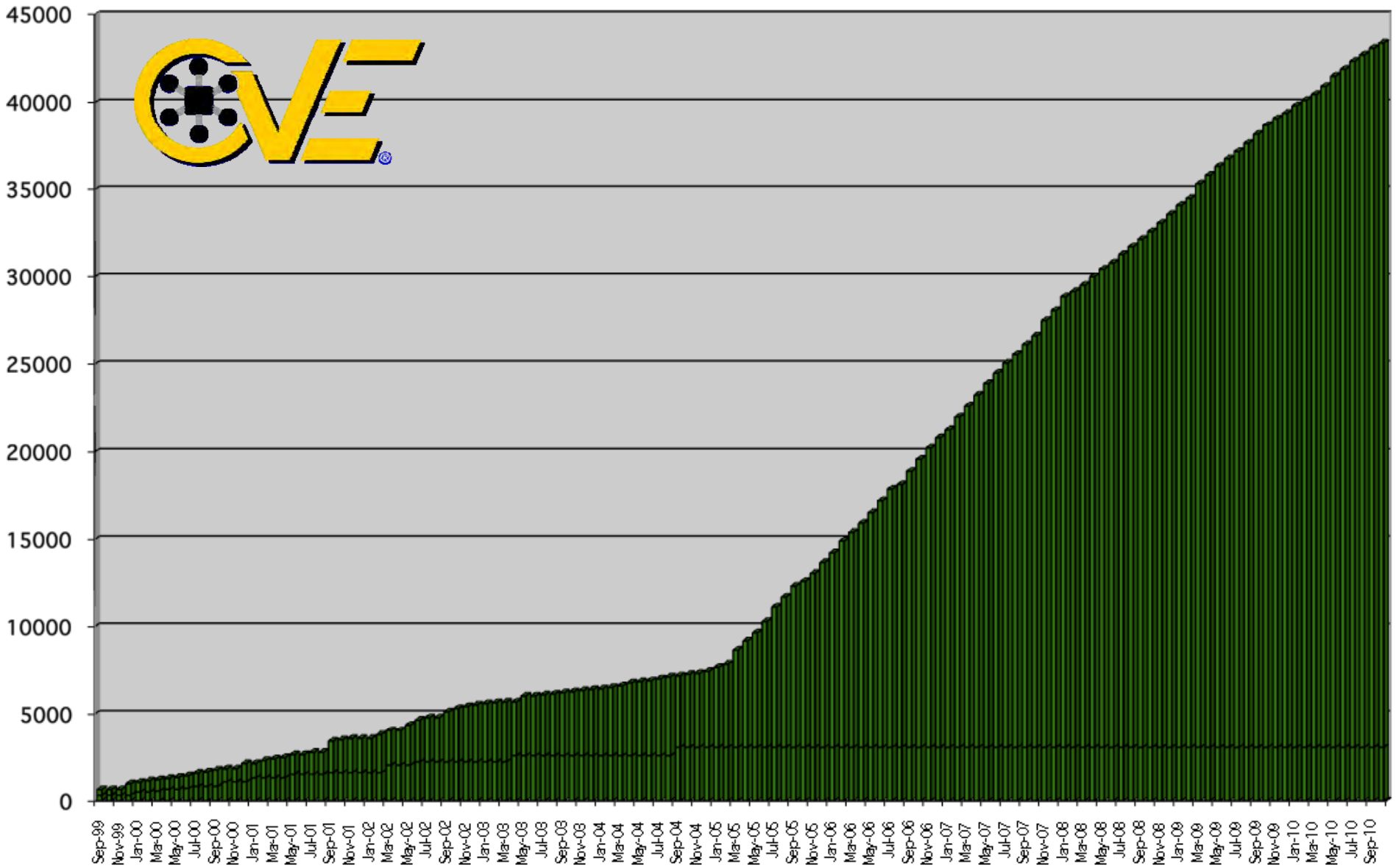


* "Scope of Supplier Expansion and Foreign Involvement" graphic in DACS www.softwaretechnews.com Secure Software Engineering, July 2005 article "Software Development Security: A Risk Management Perspective" synopsis of May 2004 GAO-04-678 report "Defense Acquisition: Knowledge of Software Suppliers Needed to Manage Risks"

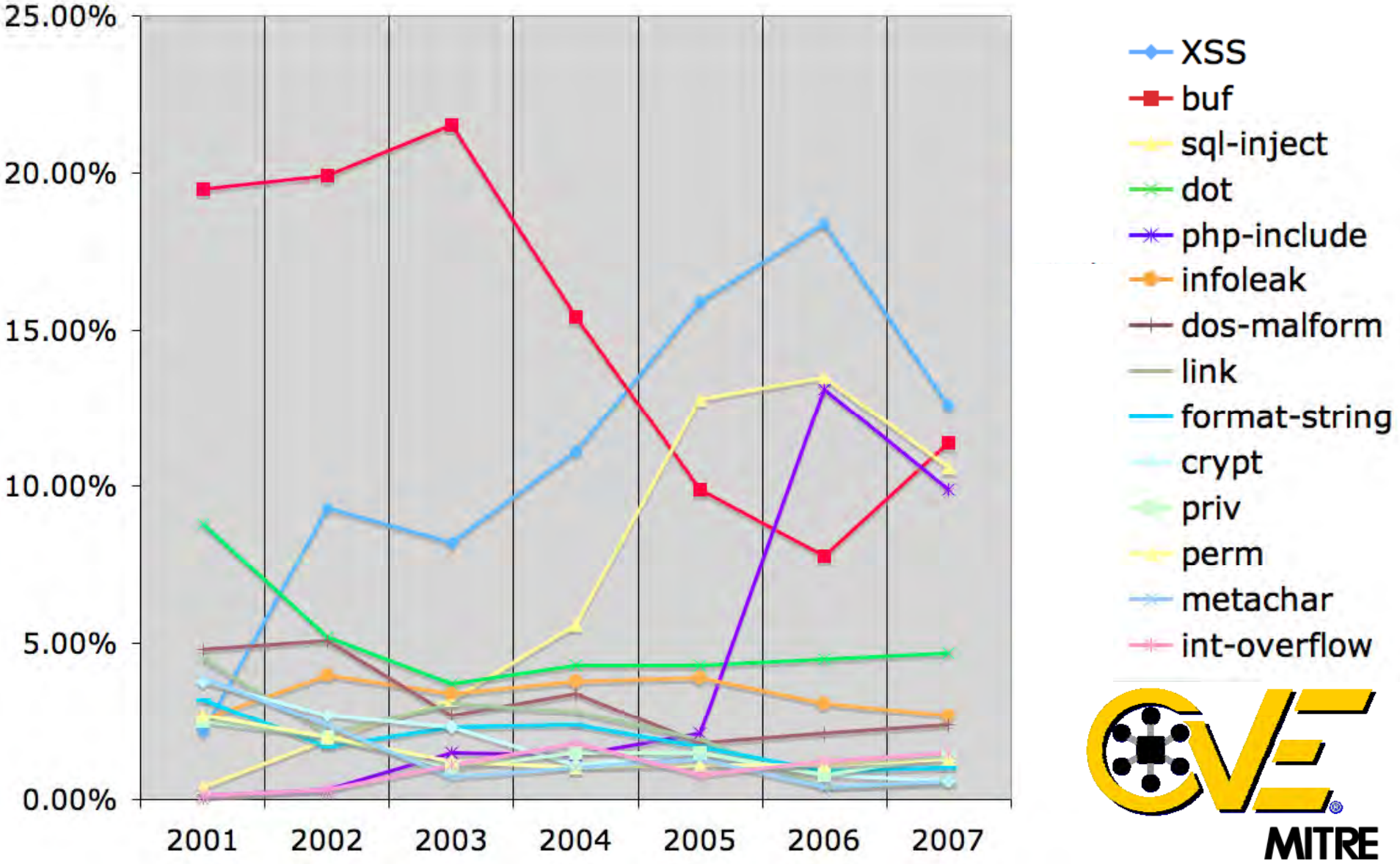
If the weaknesses in software were as easy to spot and their impact as obvious as…



CVE 1999 to 2011

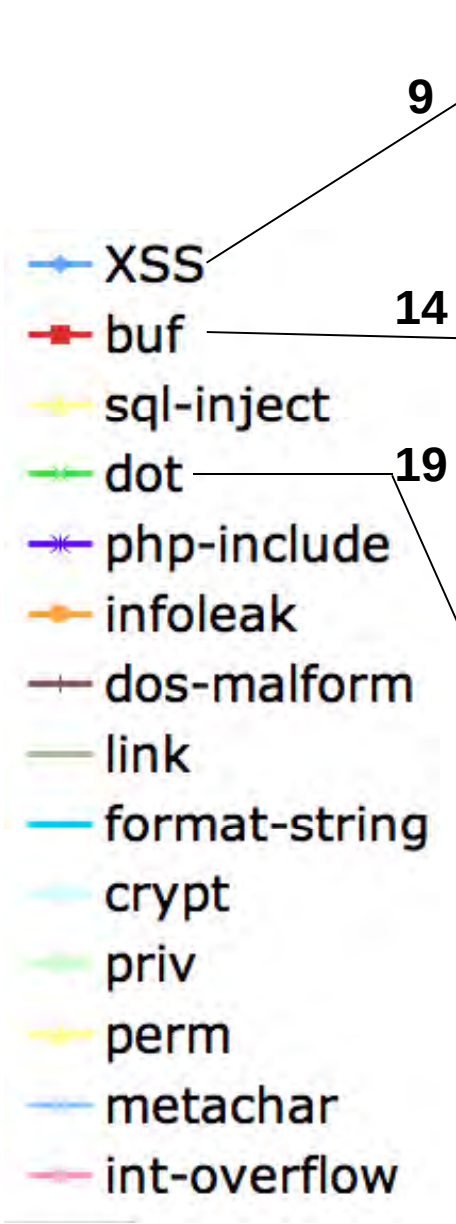


Vulnerability Type Trends:
A Look at the CVE List (2001 - 2007)



CVE
MITRE

Removing and Preventing the Vulnerabilities Requires More Specific Definitions…CWEs



9

Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting') (79)
- Improper Neutralization of Script-Related HTML Tags in a Web Page (Basic XSS) (80)
- Improper Neutralization of Script in an Error Message Web Page (81)
- Improper Neutralization of Script in Attributes of IMG Tags in a Web Page (82)
- Improper Neutralization of Script in Attributes in a Web Page (83)
- Improper Neutralization of Encoded URI Schemes in a Web Page (84)
- Doubled Character XSS Manipulations (85)
- Improper Neutralization of Invalid Characters in Identifiers in Web Pages (86)
- Improper Neutralization of Alternate XSS Syntax (87)

14

Improper Restriction of Operations within the Bounds of a Memory Buffer (119)
- Buffer Copy without Checking Size of Input ('Classic Buffer Overflow') (120)
- Write-what-where Condition (123)
- Out-of-bounds Read (125)
- Improper Handling of Length Parameter Inconsistency (130)
- Improper Validation of Array Index (129)
- Return of Pointer Value Outside of Expected Range (466)
- Access of Memory Location Before Start of Buffer (786)
- Access of Memory Location After End of Buffer (788)
- Buffer Access with Incorrect Length Value 805
- Untrusted Pointer Dereference (822)
- Use of Out-of-range Pointer Offset (823)
- Access of Uninitialized Pointer (824)
- Expired Pointer Dereference (825)

19

Path Traversal (22)
- Relative Path Traversal (23)
  - Path Traversal: '../filedir' (24)
  - Path Traversal: '/../filedir' (25)
  - <----------8 more here ---------->
  - Path Traversal: '..../' (34)
  - Path Traversal: '.../.../' (35)
- Absolute Path Traversal (36)
  - Path Traversal: '/absolute/pathname/here' (37)
  - Path Traversal: '\absolute\pathname\here' (38)
  - Path Traversal: 'C:dirname' (39)
  - Path Traversal: '\\UNC\share\name\' (Windows UNC Share) (40)

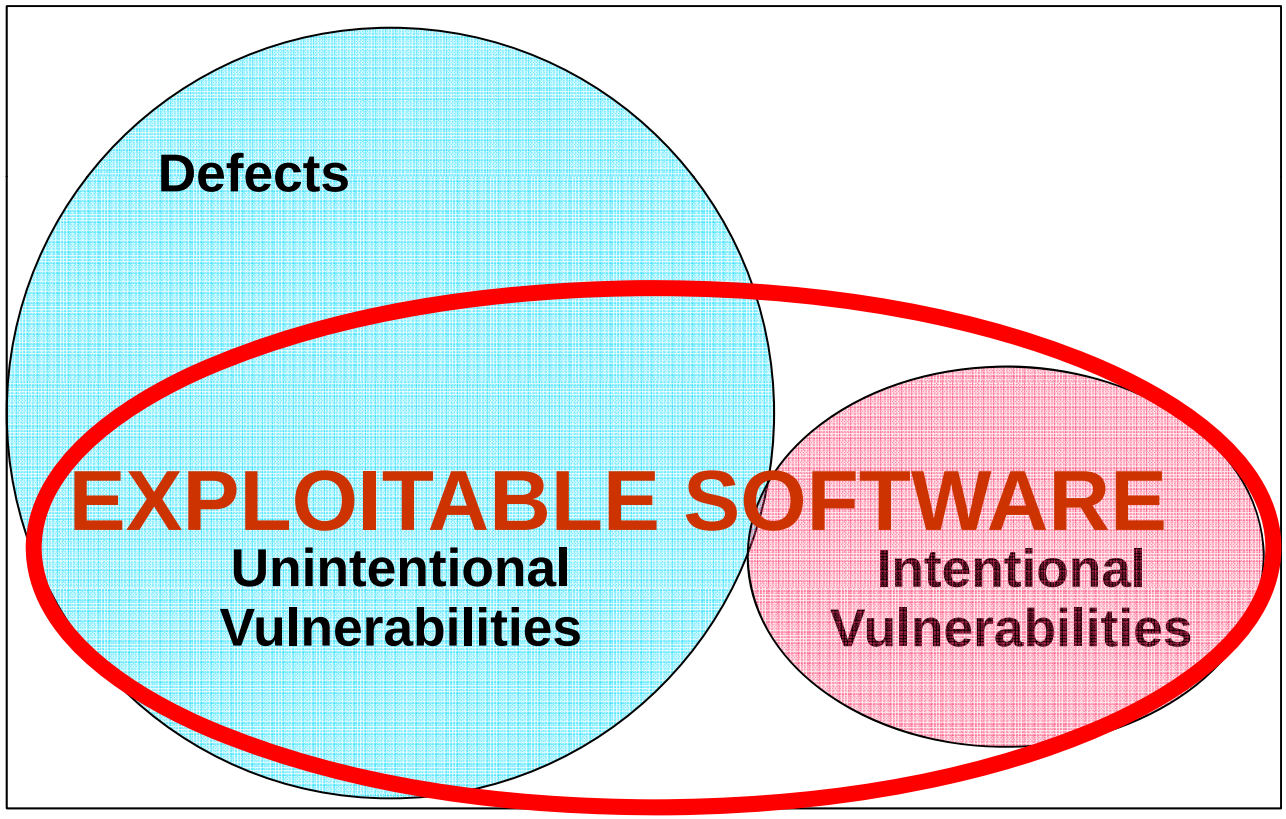
CWE™

Exploitable Software Weaknesses (a.k.a. Vulnerabilities)

Vulnerabilities can be the outcome of non-secure practices and/or malicious intent of someone in the development/support lifecycle.

The exploitation potential of a vulnerability is independent of the "intent" behind how it was introduced.

DEPLOYED SOFTWARE



Intentional vulnerabilities are spyware & malicious logic deliberately imbedded (and might not be considered defects but they can make use of the same weakness patterns as unintentional mistakes)

Note: Chart is not to scale – notional representation -- for discussions

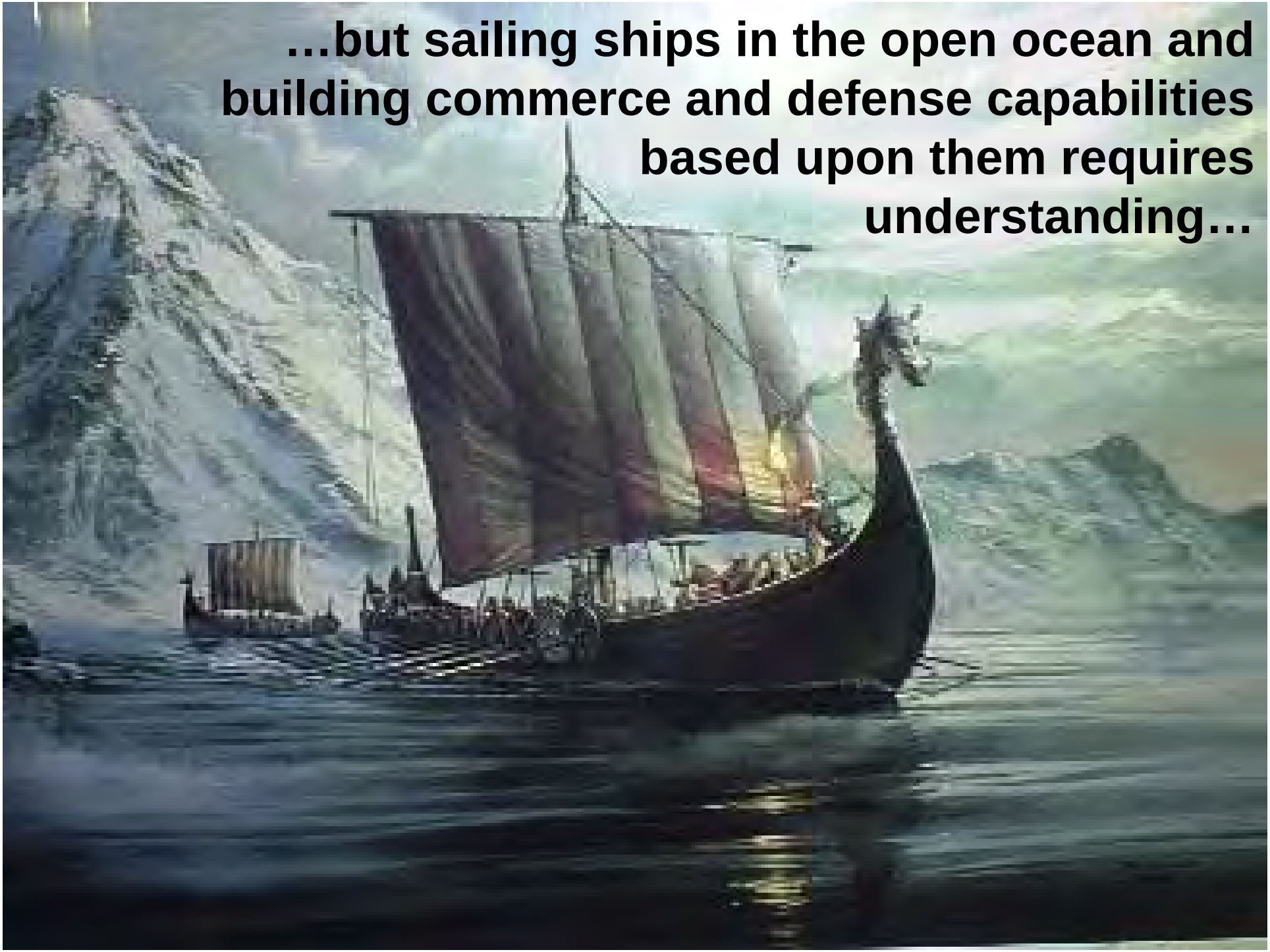
Common Weakness Enumeration (CWE)

- dictionary of weaknesses
  - weaknesses that can lead to exploitable vulnerabilities (i.e. CVEs)
  - the things we don't want in our code, design, or architecture
  - web site with XML of content, sources of content, and process used
- structured views
  - provides multiple views into CWE dictionary content
  - supports alternate views – developer/researcher/sub-views
- open community process
  - to facilitate common terms/ concepts/facts and understanding
  - allows for vendors, developers, system owners and acquirers to understand tool capabilities/ coverage and priorities
  - utilize community expertise

Foundation for other DHS, NSA, OSD, NIST, OWASP, SANS, and OMG SwA Efforts

Making Security Measurable™

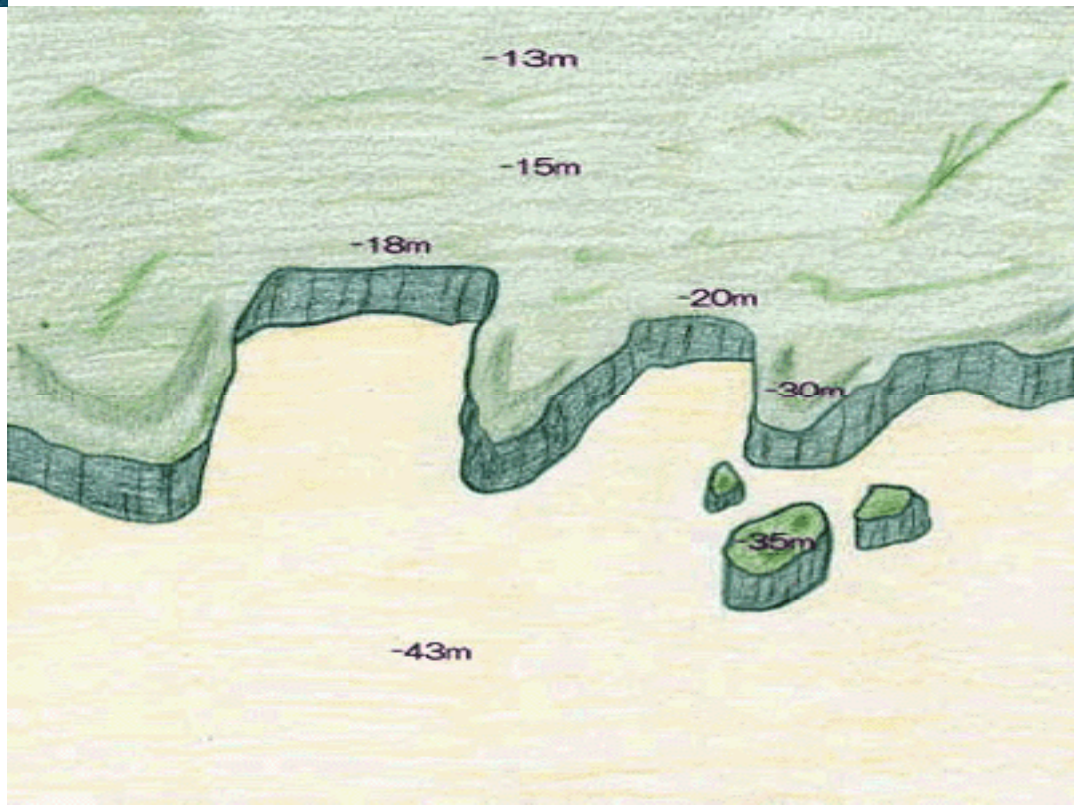

Building Software
only require a few
skills and basic
understanding…

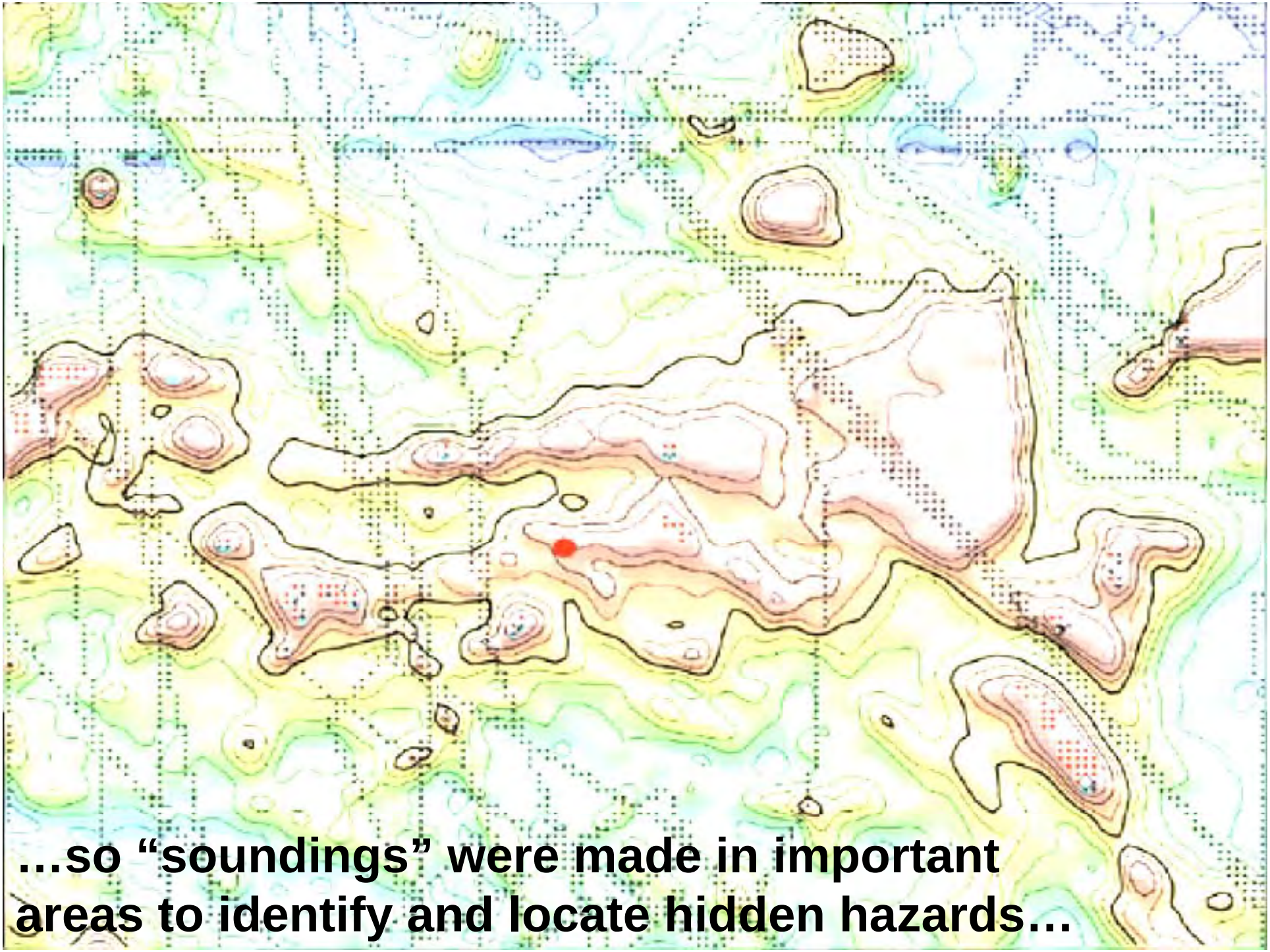

…but sailing ships in the open ocean and building commerce and defense capabilities based upon them requires understanding…


…of navigation threats…

…surface maps didn't capture the full set of threats and hazards – i.e. what was really going on…

…a more insightful depiction – one that shows what was going on under the surface – was needed…

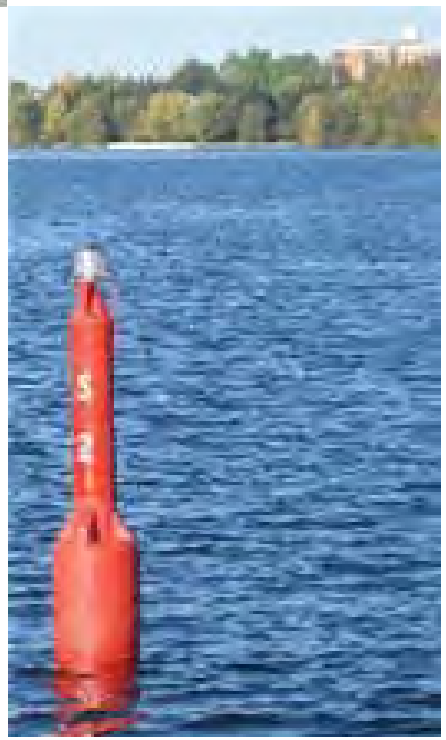




…so “soundings” were made in important areas to identify and locate hidden hazards…

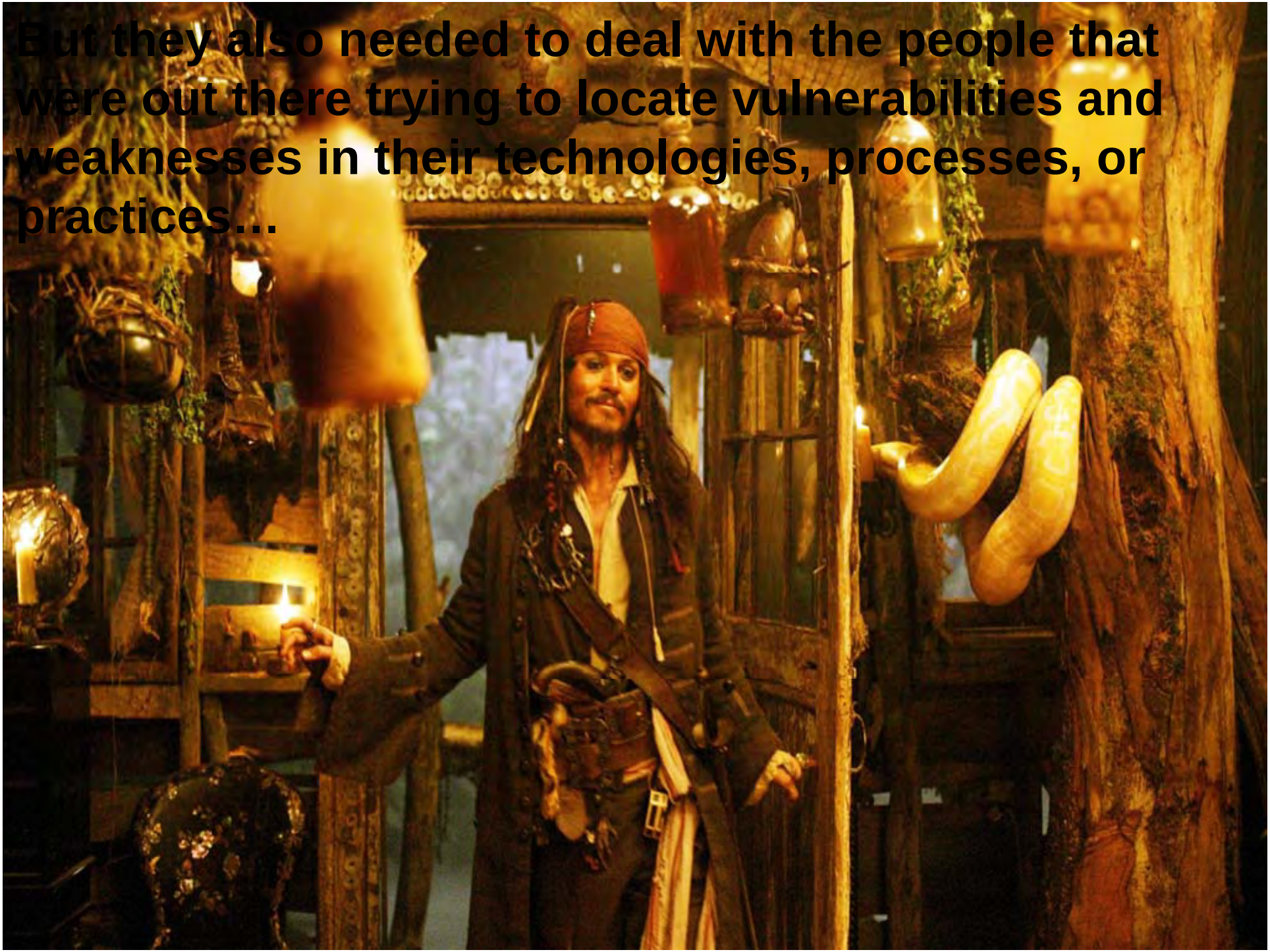
…and warning signals to help others avoid known hazards were erected along with…

…indicators showing safe ways to avoid the known hazards…

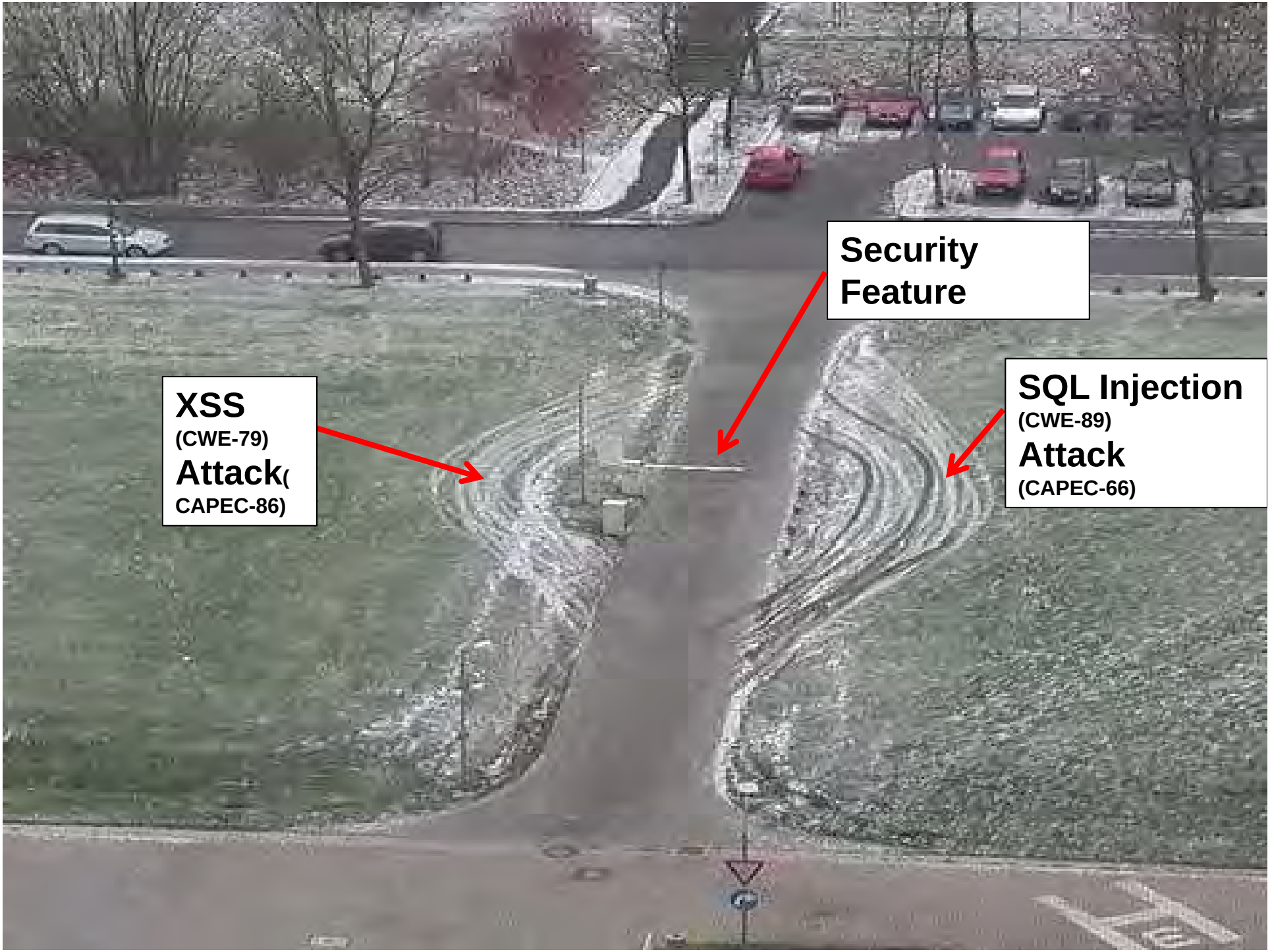



River
Know
Security
Weaknesses
Secure Coding Rules
Know
Security
Weaknesses
Know
Security
Weaknesses
Know
Security
Weaknesses
Open Waters

But they also needed to deal with the people that were out there trying to locate vulnerabilities and weaknesses in their technologies, processes, or practices…




…with defensive and offensive security capabilities.
712


Security Feature
XSS
(CWE-79)
Attack(
CAPEC-86)
SQL Injection
(CWE-89)
Attack
(CAPEC-66)

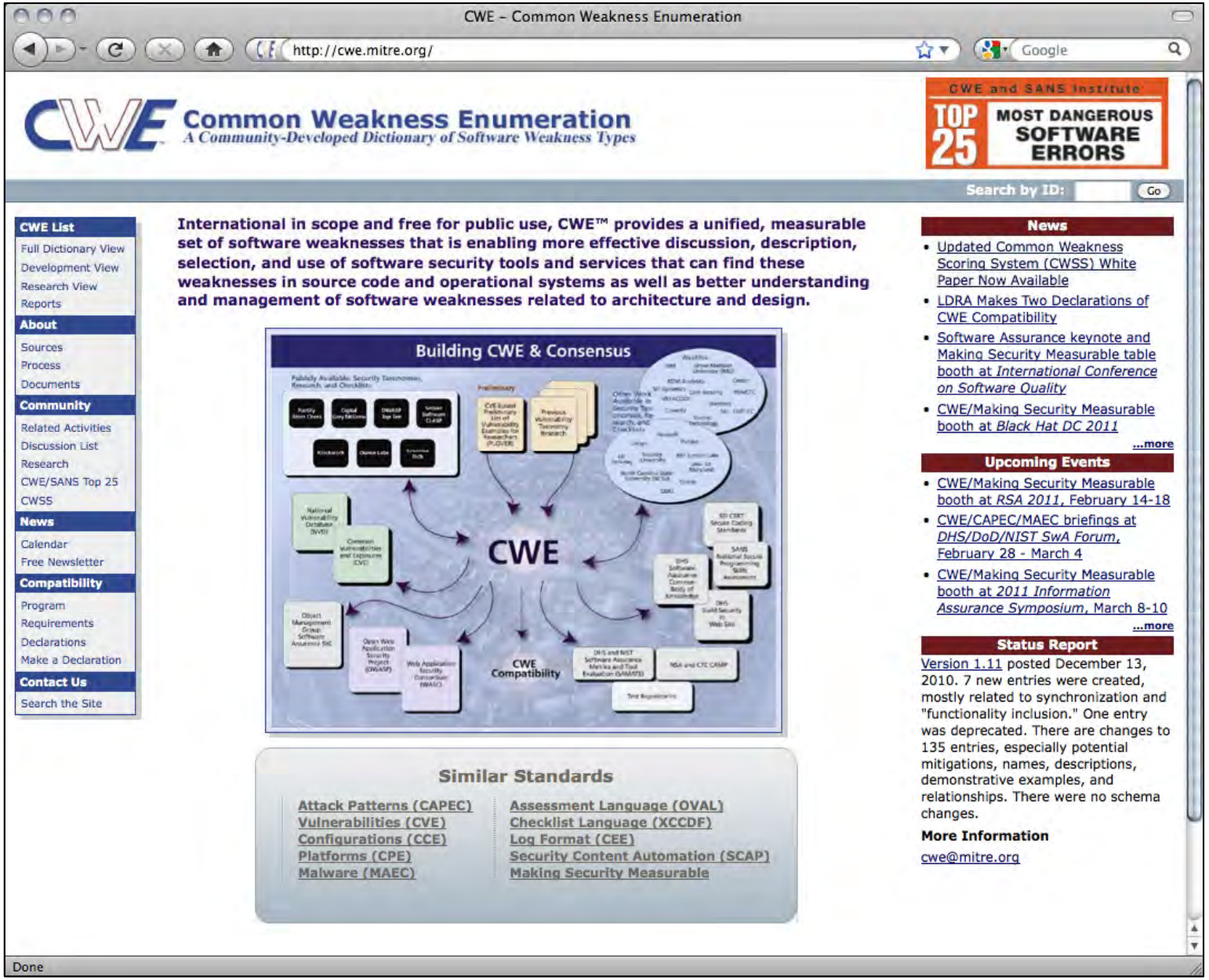
Software [In]security: Cyber Warmongering and Influence Peddling


By Gary McGraw and Ivan Arce

Nov 24, 2010

Article is provided courtesy of Addison-Wesley Professional

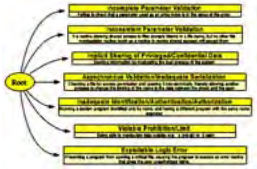
“For years in computer security, we have been attempting to protect the broken stuff from the bad people by placing a barrier between the bad people and the broken stuff. We have failed. Instead, we need to fix the broken stuff so that attacking it successfully takes far more resources and skill than is currently the case.”



Making Security Measurable™

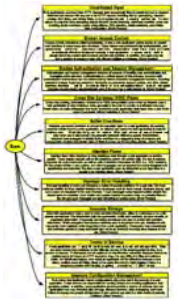
Protection Analysis

RISOS

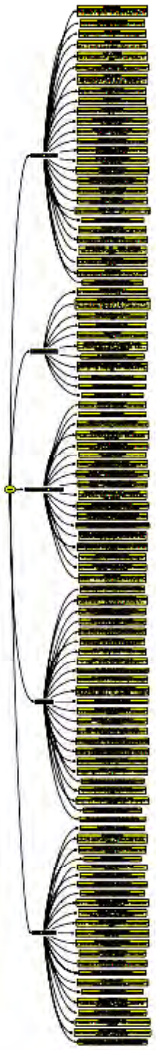


Weber

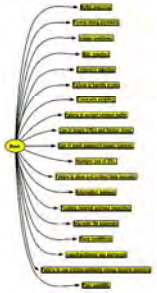
OWASP



CLASP



Microsoft

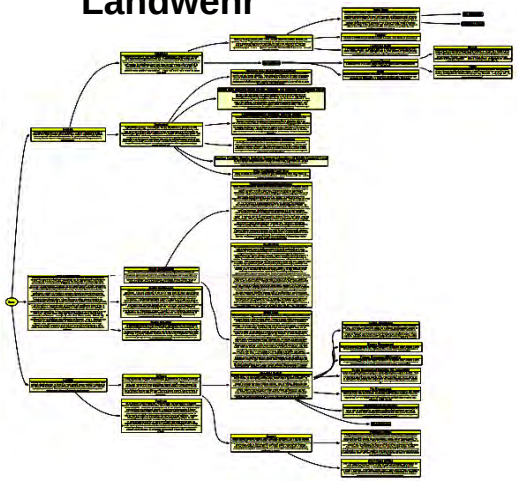


PLOVER

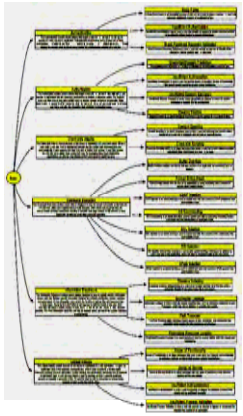
Bishop

Aslam

Landwehr



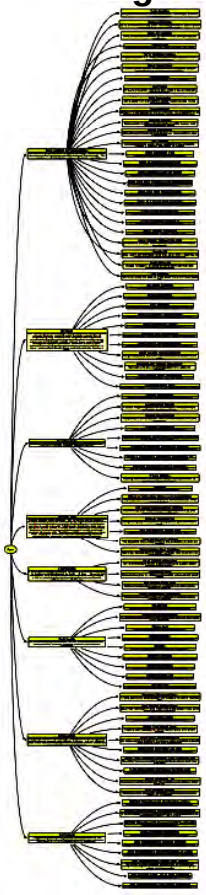
WASC

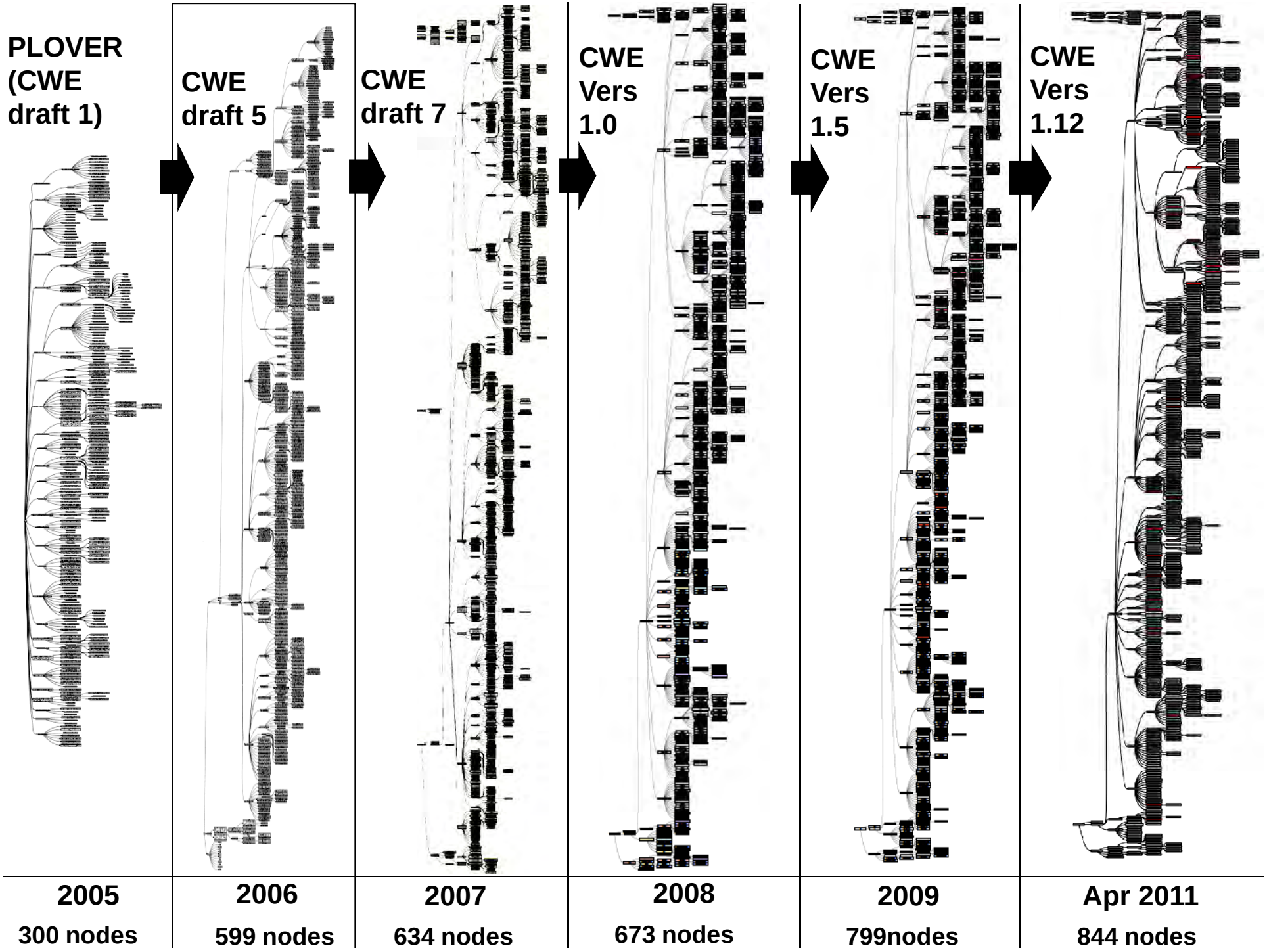


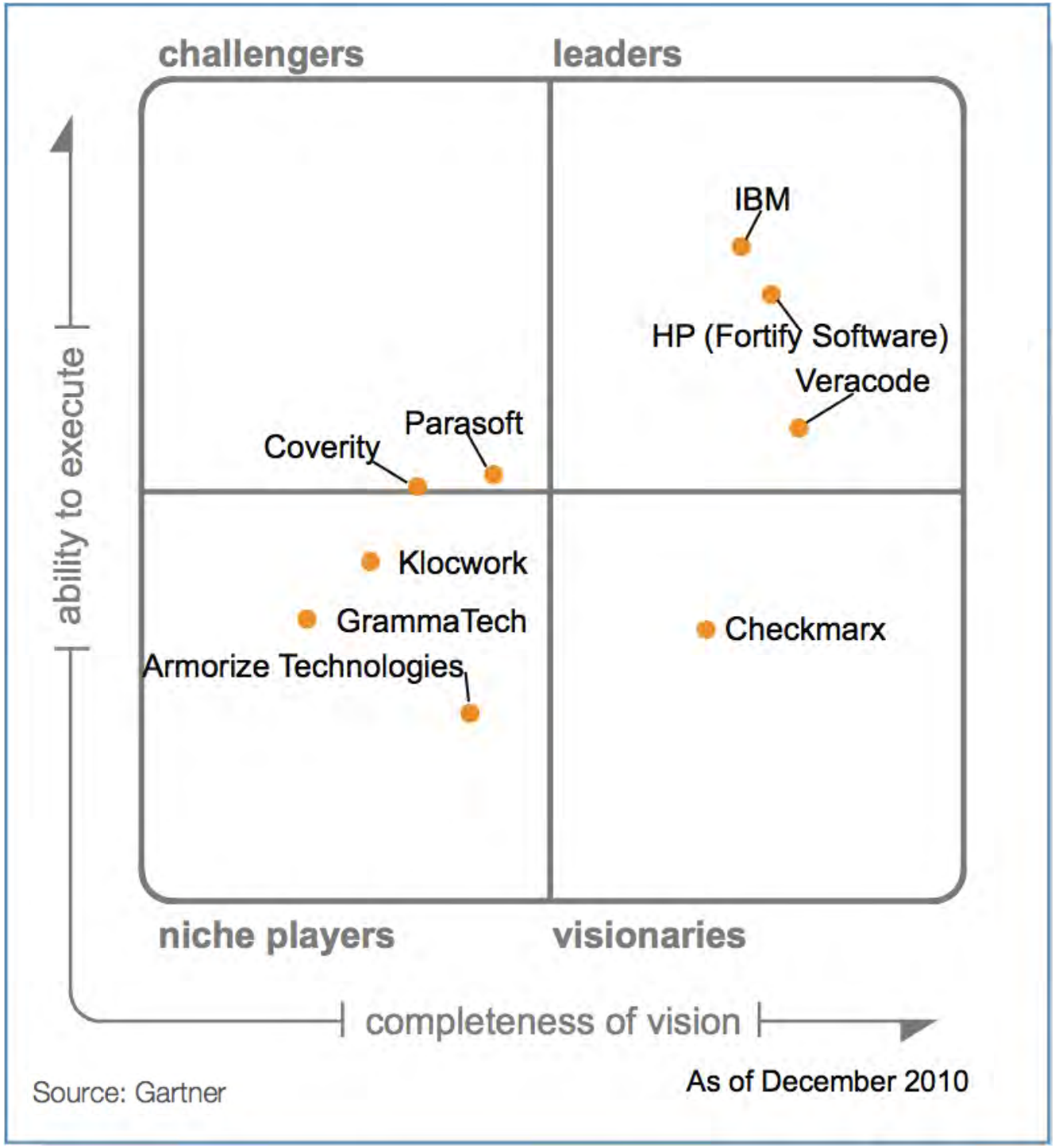
Tool A

Tool B

7 Kingdoms




PLOVER (CWE draft 1)
CWE draft 5
CWE draft 7
CWE Vers 1.0
CWE Vers 1.5
CWE Vers 1.12
2005
300 nodes
2006
599 nodes
2007
634 nodes
2008
673 nodes
2009
799nodes
Apr 2011
844 nodes



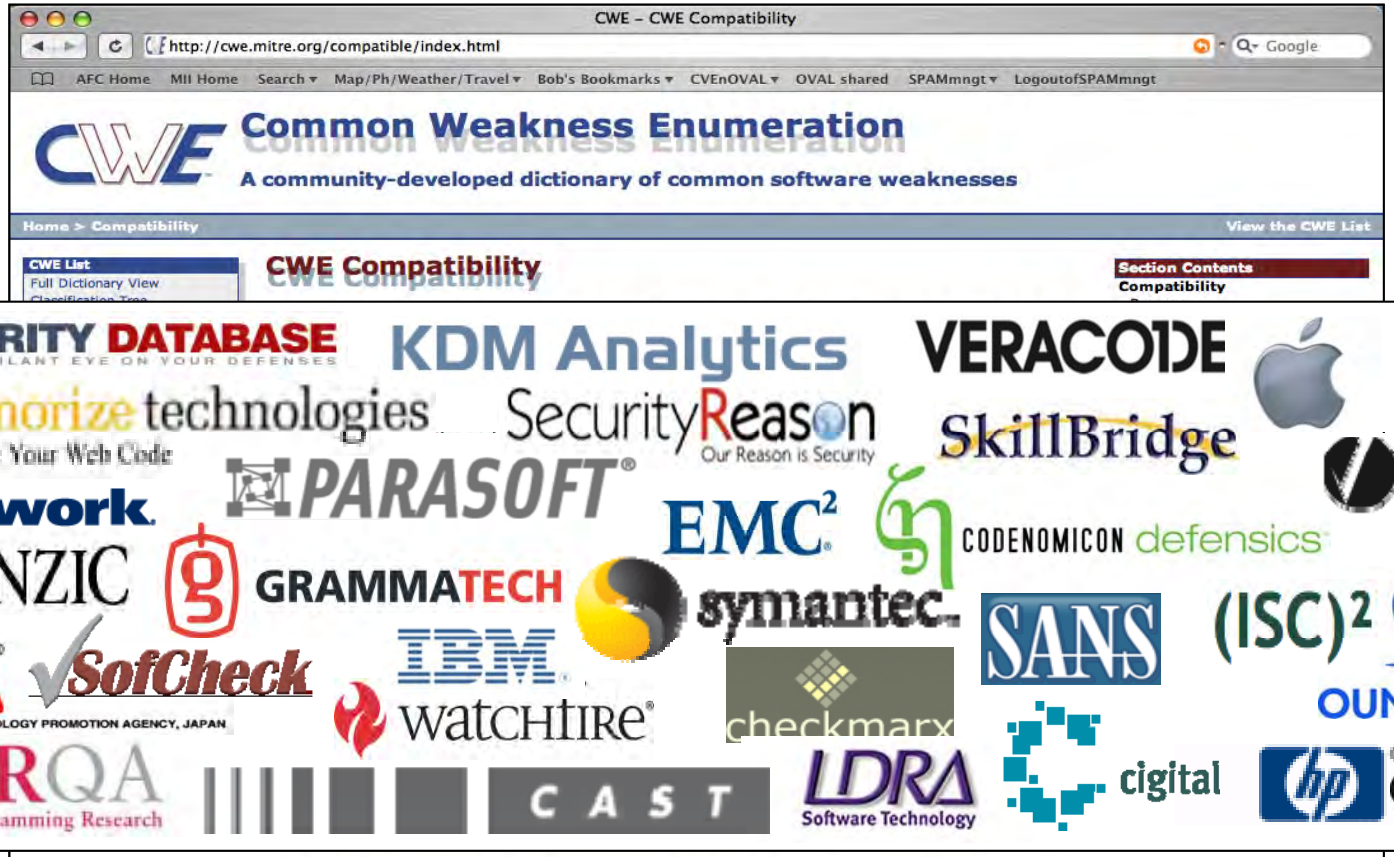
Gartner Magic Quadrant for Static Application Security Testing Tools

Plus Some Other Important Tool Players…

Cenzic
CAST Software
Polyspace
Security Innovation
LDRA
KDM Analytics
SureLogic
Programming Research Inc
SofCheck

CWE Compatibility & Effectiveness Program

(launched Feb 2007)



SECURITY DATABASE · KDM Analytics · VERACODE · FORTIFY SOFTWARE · armorize technologies · Secure Your Web Code · SecurityReason · Our Reason is Security · SkillBridge · coverity · PARASOFT · Klocwork · EMC² · CODENOMICON defensics · CENZIC · GRAMMATECH · symantec. · SANS · (ISC)² · CERIAS · IPA · INFORMATION-TECHNOLOGY PROMOTION AGENCY, JAPAN · SofCheck · IBM · watchfire · checkmarx · OUNCE LABS · PRQA Programming Research · CAST · LDRA Software Technology · cigital · hp · SPI DYNAMICS

Organizations Participating

All organizations participating in the CWE Compatibility and Effectiveness Program are listed below, including those with CWE-Compatible Products and Services and those with Declarations to Be CWE-Compatible.

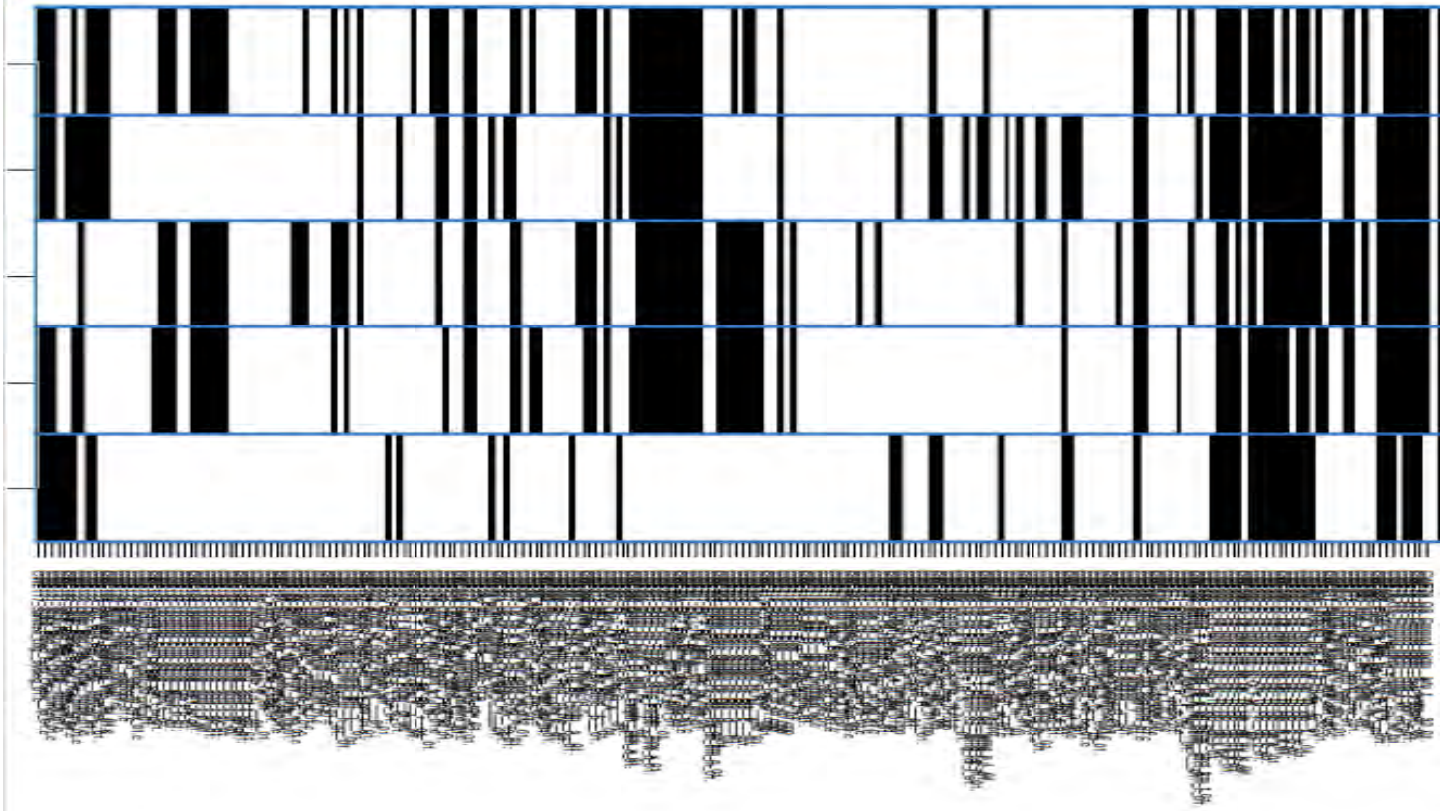
TOTALS
Organizations Participating: **29**
Products & Services: **48**

Products are listed alphabetically by organization name:

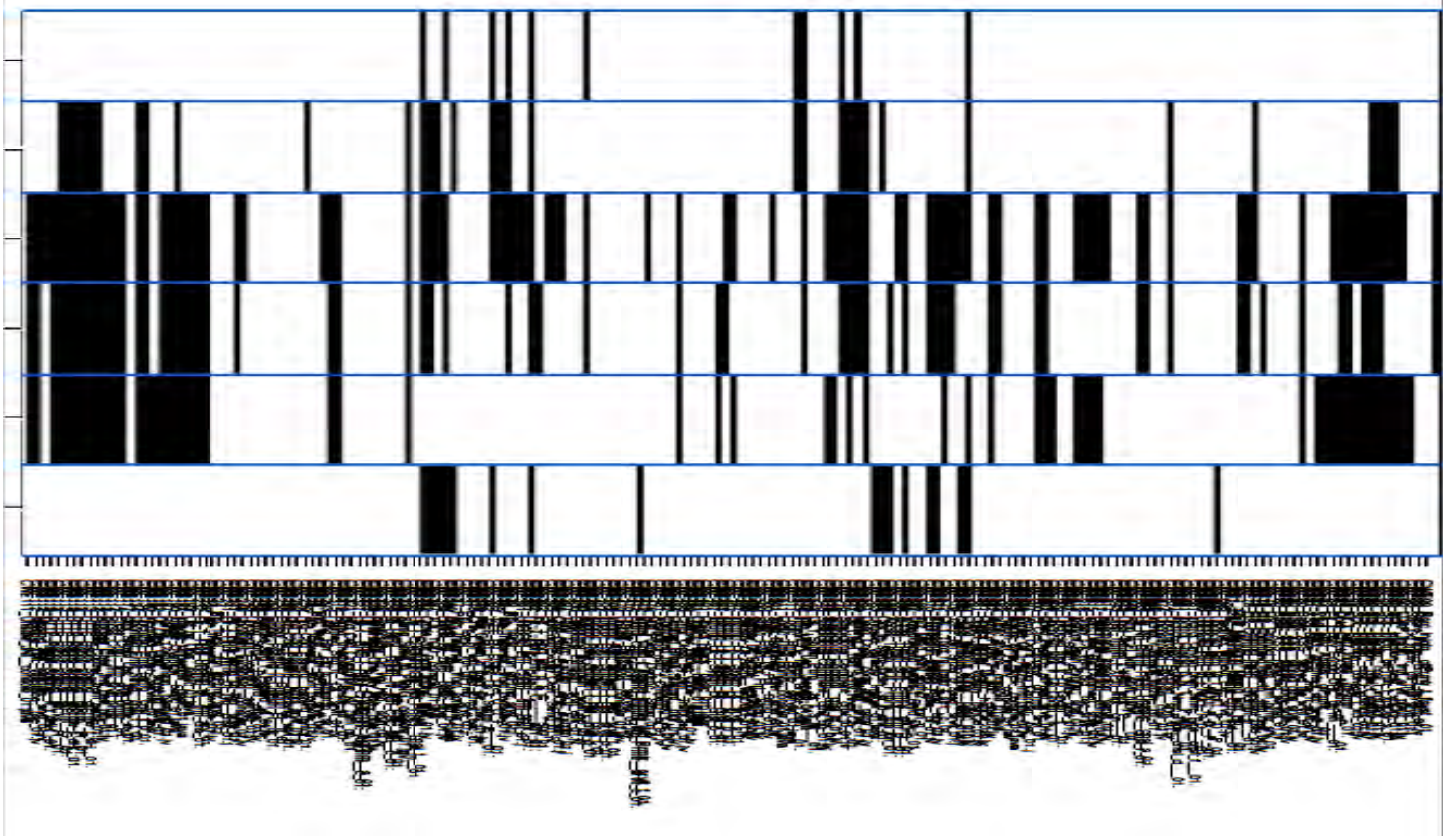
cwe.mitre.org/compatible/

December 29, 2006

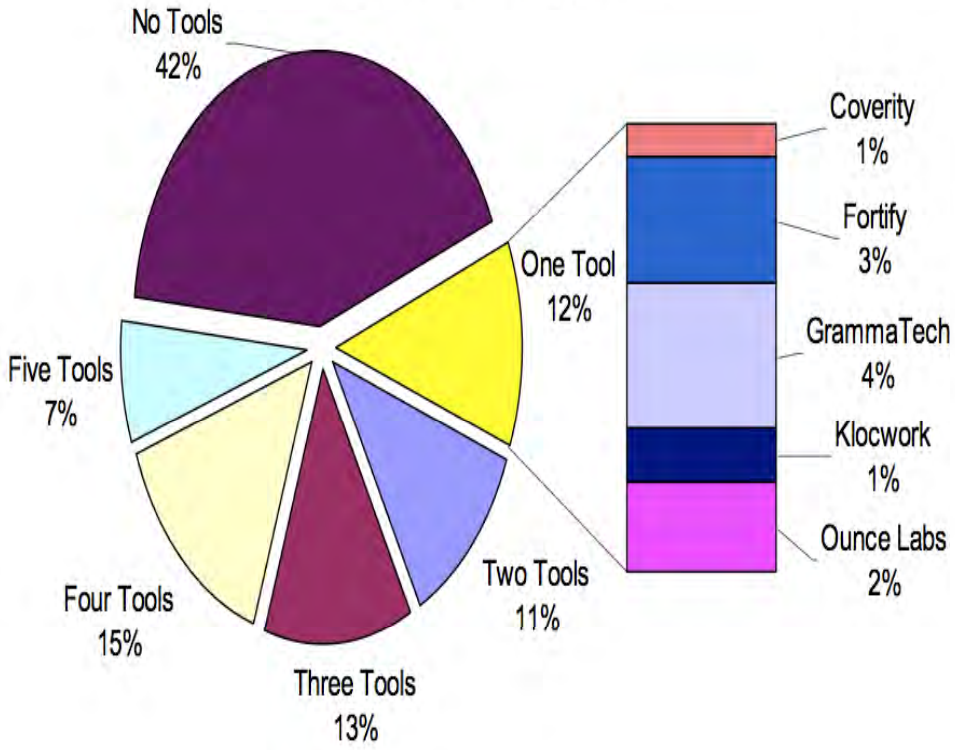
C Test Cases



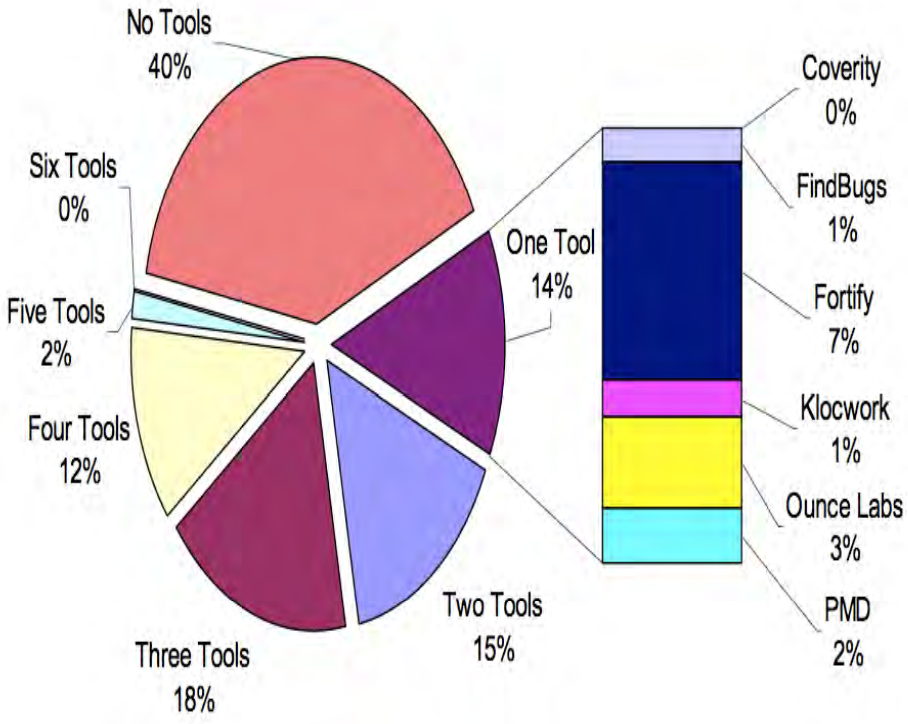
Java Test Cases



C/C++ "Breadth" Test Case Coverage



Java "Breadth" Test Case Coverage



Making Security Measurable™

Code Analysis Effectiveness Assessment…

CWE Validation
Effectiveness Testing - ?

CWE Compatibility and Effectiveness

CWEs with WhiteBox Definitions

Center For Assure SW
Tool Evaluation 2007
Tool Evaluation 2009

IARPA
STONESOUP- Securely Taking On New Executable Stuff Of Uncertain Provenance

OSD/NII
CWE Formalization

NIST SAMATE
SP 500-267
SP 500-269
SP 500-270

SAMATE Repository Dataset (SRD)

Automated Test Case Generator

NIST SATE
SATE08
SATE09
SATE10

SySA Task Force
WhiteBox Definitions-to-SBVR-to-microKDM

All of these are aimed at different aspects of understanding how well tools find CWEs in software applications and what can be done to improve that and standardize the process for expressing a tools capabilities.

CWE Coverage – Implemented…

Coverity Data Sheet

coverity

Coverity Coverage for Common Weakness Enumeration (CWE): Java

CWE ID	Coverity Static Analysis Checker
171	BAD_EQ
252	CHECKED_RETURN
366	GUARDED_BY_VIOLATION
	INDIRECT_GUARDED_BY_VIOLATION
	NON_STATIC_GUARDING_STATIC
	VOLATILE_ATOMICITY
382	DC.CODING_STYLE
	BAD_OVERRIDE
	DC.EXPLICIT_DEPRECATION
	DC.GC
	MUTABLE_COMPARISON
398	MUTABLE_HASHCODE

Coverity Data Sheet

coverity

Coverity Coverage For Common Weakness Enumeration (CWE): C/C++

CWE ID	Coverity Static Analysis Checker	Checker Description	Type of Security Risk
	TAINTED_SCALAR	Use of untrusted scalar value	
		Untrusted value as an argument	Alter control flow
		Use of untrusted value	Arbitrary control of a resour
		Use of untrusted string value	Arbitrary code execution
		User pointer dereference	
		Out-of-bounds access	
		Stray pointer arithmetic	
		COM bad conversion to BSTR	
		Overflowed array index write	
		Overflowed pointer write	
		Using invalid iterator	Arbitrary code execution
		Iterator container mismatch	Alter control flow
		Splice iterator mismatch	Read sensitive information
		Allocation size error	Denial of service
		Out-of-bounds access	
		Out-of-bounds write	
		Out-of-bounds access	
		Out-of-bounds write	
		Argument cannot be negative	
		Copy into fixed size buffer	
		Destination buffer too small	
		Possible buffer overflow	
		Allocation too small for type	Unauthorized code executio
		Buffer overflow	Denial of service
		Copy into fixed size buffer	
		Destination buffer too small	
		Unbounded source buffer	

CENZIC

Cenzic Product Suite is CWE Compatible

Cenzic Hailstorm Enterprise ARC, Cenzic Hailstorm Professional and Cenzic ClickToSecure are compatible with the CWE standard or Common Weakness Enumeration as maintained by MITRE Corporation. Web security assessment results from the Hailstorm product suite are mapped to the relevant CWE ID's providing users with additional information to clearly and describe common weaknesses found in Web applications.

For additional details on CWE, please visit: http://cwe.mitre.org/data/index.html

The following is a mapping between Cenzic's SmartAttacks and CWE ID's:

	Cenzic SmartAttack Name	CWE ID's
1	Application Exception	CWE-388: Error Handling
2	Application Exception (VIS)	CWE-388: Error Handling
3	Application Path Disclosure	CWE-200: Information Leak (rough match)
4	Authentication Bypass	CWE-89: Failure to Sanitize Data into SQL Queries (aka 'SQL Injection') (rough match)
5	Authorization Bypass	CWE-285: Missing or Inconsistent Access Control, CWE-425: Direct Request ('Forced Browsing')
6	Blind SQL Injection	CWE-89: Failure to Sanitize Data into SQL Queries (aka 'SQL Injection')
7	Blind SQL Injection (VIS)	CWE-89: Failure to Sanitize Data into SQL Queries (aka 'SQL Injection')
8	Browse HTTP from HTTPS Url	CWE-200: Information Leak
9	Brute Force Login	CWE-521: Weak Password Requirements
10	Buffer Overflow	CWE-120: Unbounded Transfer ('Classic Buffer Overflow')
11	Buffer Overflow (VIS)	CWE-120: Unbounded Transfer ('Classic Buffer Overflow')
12	Check Basic Auth over HTTP	CWE-200: Information Leak
13	Check HTTP Methods	CWE-650: Trusting HTTP Permission Methods on the Server Side

Cenzic CWE Brochure | October 2009

CWE IDs mapped to Klocwork Java issue types - current

CWE IDs mapped to Klocwork Java issue types

From current

CWE IDs mapped to Klocwork Java issue types

See also Detected Java Issues.

CWE IDs mapped to Klocwork C and C++ issue types/ja -...

CWE IDs mapped to Klocwork C and C++ issue types/ja

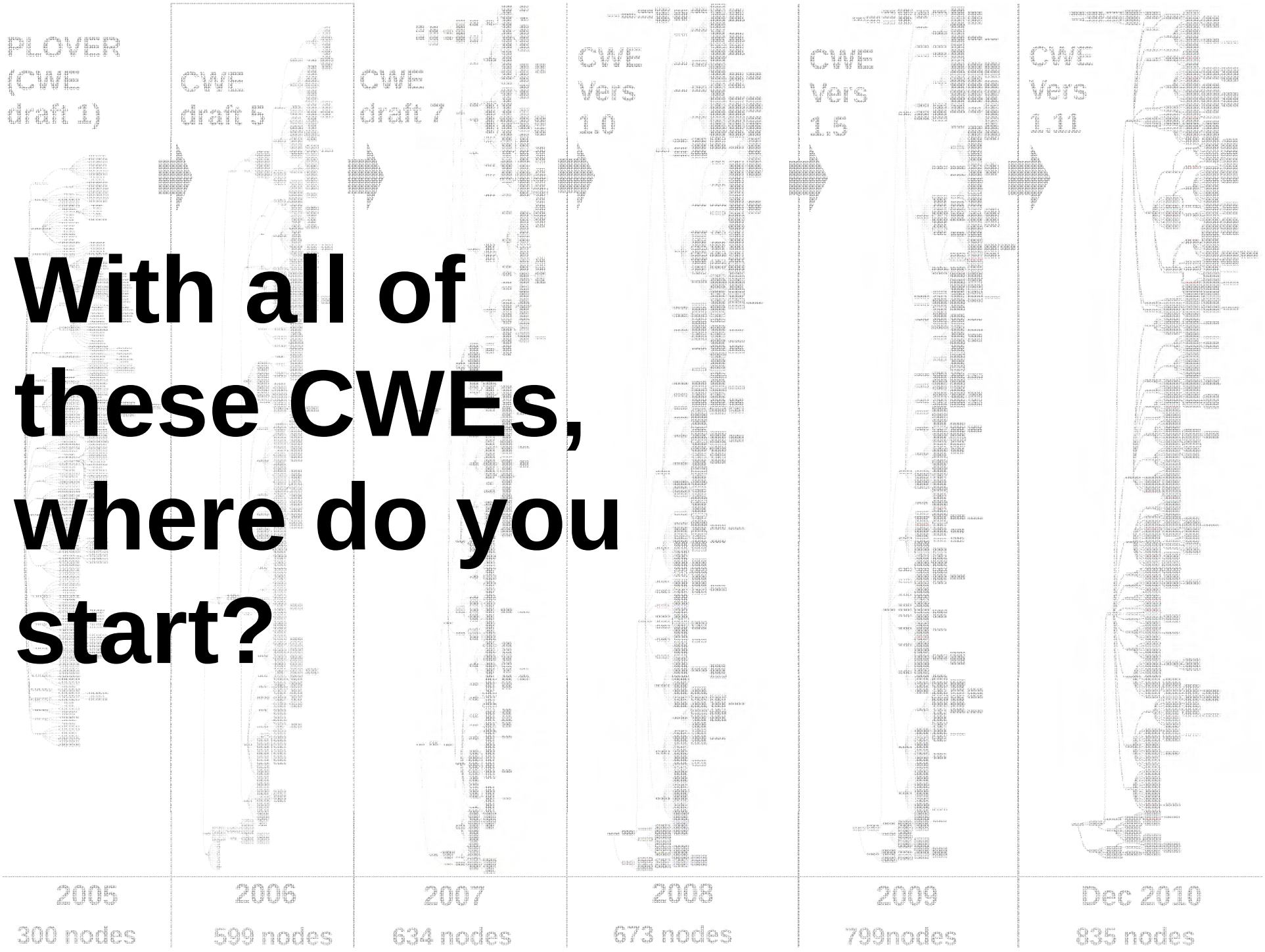
From current

< CWE IDs mapped to Klocwork C and C++ issue types
CWE IDs mapped to Kocwork C and C++ issue types/ja

その他の情報 Detected C and C++ Issues.

CWE ID	説明
20 (http://cwe.mitre.org/data/definitions/20.html)	ABV.TAINTED 未検証入力によるバッファ オーバーフロー SV.TAINTED.GENERIC 未検証文字列データの使用 SV.TAINTED.ALLOC_SIZE メモリ割り当てにおける未検証の整数の使用 SV.TAINTED.CALL.INDEX_ACCESS =関数呼び出しにおける未検証整数の配列インデックスとしての使用
22 (http://cwe.mitre.org/data/definitions/22.html)	SV.CUDS.MISSING_ABSOLUTE_PATH ファイルのロードでの絶対パスの不使用
73 (http://cwe.mitre.org/data/definitions/73.html)	SV.CUDS.MISSING_ABSOLUTE_PATH ファイルのロードでの絶対パスの不使用
74 (http://cwe.mitre.org/data/definitions/74.html)	SV.TAINTED.INJECTION コマンド インジェクション
77 (http://cwe.mitre.org/data/definitions/77.html)	SV.CODE_INJECTION.SHELL_EXEC シェル実行へのコマンド インジェクション
78 (http://cwe.mitre.org/data/definitions/78.html)	NNTS.TAINTED 未検証ユーザ入力が原因のバッファ オーバーフロー - 非 NULL 終端文字列 SV.TAINTED.INJECTION コマンド インジェクション
88 (http://cwe.mitre.org	SV.TAINTED.INJECTION コマンド インジェクション NNTS.TAINTED 未検証ユーザ入力が原因のバッファ オーバーフロー

1 of 7



With all of these CWEs, where do you start?

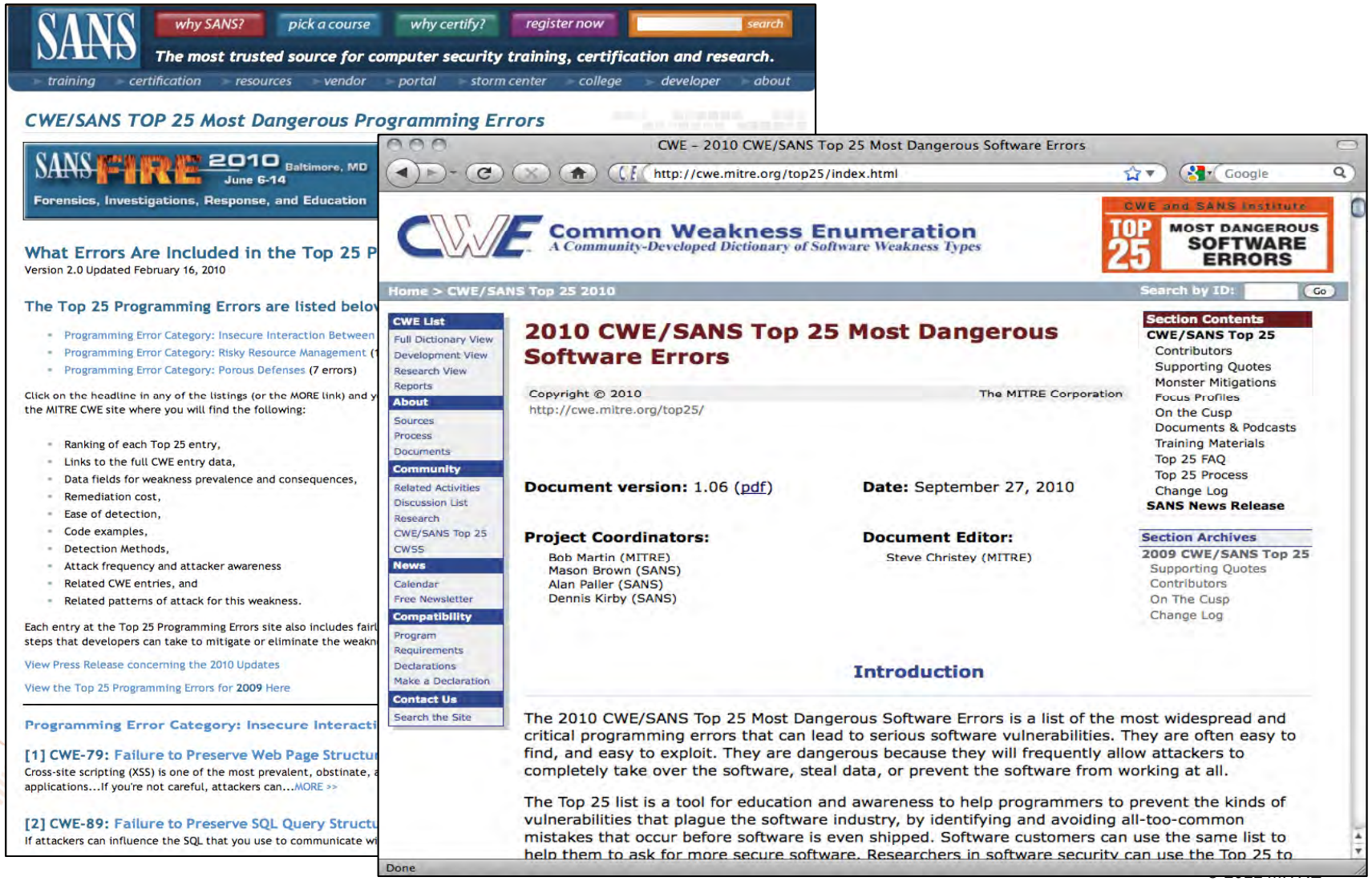
2009 SANS/CWE Top 25 Programming Errors
(released 12 Jan 2009) cwe.mitre.org/top25/

- List selected by security experts from 34 organizations

Making Security Measurable™

2010 CWE/SANS Top 25 Programming Errors
(released 16 Feb 2010) cwe.mitre.org/top25/

- List selected by security experts from 34 organizations



Making Security Measurable™

Main Goals

- Raise awareness for developers
- Help universities to teach secure coding
- Empower customers who want to ask for more secure software
- Provide a starting point for in-house software shops to measure their own progress

Making Security Measurable™


Robert C. Seacord
CERT
Ryan Barnett
Pascal Meunier
CERIAS, Purdue University
Antonio Fontes
Matt Bishop
Kenneth van Wyk
Masato Terada
Sean Barnum
Mahesh Saptarshi
Cassio Goldschmidt
Adam Hahn
Jeff Williams
Carsten Eiram
Josh Drake
Chuck Willis
Michael Howard
Bruce Lowenthal
Mark J. Cox
Jacob West
Djenana Campara
James Walden
Frank Kim
Chris Eng
Chris Wysopal
Veracode, Inc.
2009
CWE – Top 25 Credited Contributors
http://cwe.mitre.org/top25/contributors.html
Common Weakness Enumeration
A Community-Developed Dictionary of Software Weakness Types
2010
Home > CWE/SANS Top 25 > Credited Contributors
Section Contents
CWE/SANS Top 25
Contributors
Supporting Quotes
Monster Mitigations
Focus Profiles
On the Cusp
Documents & Podcasts
Training Materials
Top 25 FAQ
Top 25 Process
Change Log
SANS News Release
Section Archives
2009 CWE/SANS Top 25
Supporting Quotes
Contributors
On The Cusp
Change Log
CERIAS
Center for Education and Research in Information Assurance and Security
Microsoft
ORACLE
cigital
Sun microsystems
EMC²
redhat
Secunia
Stay Secure
SRI International
NSA
DHS
IPA
INFORMATION-TECHNOLOGY PROMOTION AGENCY, JAPAN
SANS
GRAMMATECH
SRA
FORTIFY SOFTWARE
symantec.
armorize technologies
Secure Your Web Code
VERACODE
McAfee
Juniper NETWORKS
Homeland Security
CWE is a Software Assurance strategic initiative sponsored by the National Cyber Security Division of the U.S. Department of Homeland Security.
This Web site is hosted by The MITRE Corporation.
Copyright 2010, The MITRE Corporation. CWE and the CWE logo are trademarks of The MITRE Corporation.
Contact cwe@mitre.org for more information.
Privacy policy
Terms of use
Contact us

Insecure Interaction Between Components

These weaknesses are related to insecure ways in which data is sent and received between separate components, modules, programs, processes, threads, or systems.

- CWE-20: Improper Input Validation
- CWE-116: Improper Encoding or Escaping of Output
- CWE-89: Failure to Preserve SQL Query Structure (aka 'SQL Injection')
- CWE-79: Failure to Preserve Web Page Structure (aka 'Cross-site Scripting')
- CWE-78: Failure to Preserve OS Command Structure (aka 'OS Command Injection')
- CWE-319: Cleartext Transmission of Sensitive Information
- CWE-352: Cross-Site Request Forgery (CSRF)
- CWE-362: Race Condition
- CWE-209: Error Message Information Leak

Risky Resource Management

The weaknesses in this category are related to ways in which software does not properly manage the creation, usage, transfer, or destruction of important system resources.

- CWE-119: Failure to Constrain Operations within the Bounds of a Memory Buffer
- CWE-642: External Control of Critical State Data
- CWE-73: External Control of File Name or Path
- CWE-426: Untrusted Search Path
- CWE-94: Failure to Control Generation of Code (aka 'Code Injection')
- CWE-494: Download of Code Without Integrity Check
- CWE-404: Improper Resource Shutdown or Release
- CWE-665: Improper Initialization
- CWE-682: Incorrect Calculation

Porous Defenses

The weaknesses in this category are related to defensive techniques that are often misused, abused, or just plain ignored.

- CWE-285: Improper Access Control (Authorization)
- CWE-327: Use of a Broken or Risky Cryptographic Algorithm
- CWE-259: Hard-Coded Password
- CWE-732: Insecure Permission Assignment for Critical Resource
- CWE-330: Use of Insufficiently Random Values
- CWE-250: Execution with Unnecessary Privileges
- CWE-602: Client-Side Enforcement of Server-Side Security

Insecure Interaction Between Components

These weaknesses are related to insecure ways in which data is sent and received between separate components, modules, programs, processes, threads, or systems.

For each weakness, its ranking in the general list is provided in square brackets.

Rank	CWE ID	Name
[1]	CWE-79	Failure to Preserve Web Page Structure ('Cross-site Scripting')
[2]	CWE-89	Improper Sanitization of Special Elements used in an SQL Command ('SQL Injection')
[4]	CWE-352	Cross-Site Request Forgery (CSRF)
[8]	CWE-434	Unrestricted Upload of File with Dangerous Type
[9]	CWE-78	Improper Sanitization of Special Elements used in an OS Command ('OS Command Injection')
[17]	CWE-209	Information Exposure Through an Error Message
[23]	CWE-601	URL Redirection to Untrusted Site ('Open Redirect')
[25]	CWE-362	Race Condition

Risky Resource Management

The weaknesses in this category are related to ways in which software does not properly manage the creation, usage, transfer, or destruction of important system resources.

Rank	CWE ID	Name
[3]	CWE-120	Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')
[7]	CWE-22	Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
[12]	CWE-805	Buffer Access with Incorrect Length Value
[13]	CWE-754	Improper Check for Unusual or Exceptional Conditions
[14]	CWE-98	Improper Control of Filename for Include/Require Statement in PHP Program ('PHP File Inclusion')
[15]	CWE-129	Improper Validation of Array Index
[16]	CWE-190	Integer Overflow or Wraparound
[18]	CWE-131	Incorrect Calculation of Buffer Size
[20]	CWE-494	Download of Code Without Integrity Check
[22]	CWE-770	Allocation of Resources Without Limits or Throttling

Porous Defenses

The weaknesses in this category are related to defensive techniques that are often misused, abused, or just plain ignored.

Rank	CWE ID	Name
[5]	CWE-285	Improper Access Control (Authorization)
[6]	CWE-807	Reliance on Untrusted Inputs in a Security Decision
[10]	CWE-311	Missing Encryption of Sensitive Data
[11]	CWE-798	Use of Hard-coded Credentials
[19]	CWE-306	Missing Authentication for Critical Function
[21]	CWE-732	Incorrect Permission Assignment for Critical Resource
[24]	CWE-327	Use of a Broken or Risky Cryptographic Algorithm

2 CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')

Summary

Weakness Prevalence	High	Consequences	Data loss, Security bypass
Remediation Cost	Low	Ease of Detection	Easy
Attack Frequency	Often	Attacker Awareness	High

Discussion

These days, it seems as if software is all about the data: getting it into the database, pulling it from the database, massaging it into information, and sending it elsewhere for fun and profit. If attackers can influence the SQL that you use to communicate with your database, then suddenly all your fun and profit belongs to them. If you use SQL queries in security controls such as authentication, attackers could alter the logic of those queries to bypass security. They could modify the queries to steal, corrupt, or otherwise change your underlying data. They'll even steal data one byte at a time if they have to, and they have the patience and know-how to do so.

Technical Details | *Code Examples* | *Detection Methods* | *References*

Prevention and Mitigations

Architecture and Design
Use a vetted library or framework that does not allow this weakness to occur or provides constructs that make this weakness easier to avoid.
For example, consider using persistence layers such as Hibernate or Enterprise Java Beans, which can provide significant protection against SQL injection if used properly.

Architecture and Design
If available, use structured mechanisms that automatically enforce the separation between data and code. These mechanisms may be able to provide the relevant quoting, encoding, and validation automatically, instead of relying on the developer to provide this capability at every point where output is generated.
Process SQL queries using prepared statements, parameterized queries, or stored procedures. These features should accept parameters or variables and support strong typing. Do not dynamically construct and execute query strings within these features using "exec" or similar functionality, since you may

Monster Mitigations

These mitigations will be effective in eliminating or reducing the severity of the Top 25. These mitigations will also address many weaknesses that are not even on the Top 25. If you adopt these mitigations, you are well on your way to making more secure software.

A Monster Mitigation Matrix is also available to show how these mitigations apply to weaknesses in the Top 25.

ID	Description
M1	Establish and maintain control over all of your inputs.
M2	Establish and maintain control over all of your outputs.
M3	Lock down your environment.
M4	Assume that external components can be subverted, and your code can be read by anyone.
M5	Use industry-accepted security features instead of inventing your own.
GP1	(general) Use libraries and frameworks that make it easier to avoid introducing weaknesses.
GP2	(general) Integrate security into the entire software development lifecycle.
GP3	(general) Use a broad mix of methods to comprehensively find and prevent weaknesses.
GP4	(general) Allow locked-down clients to interact with your software.

M1	M2	M3	M4	M5	CWE
High		DiD	Mod		CWE-22: Improper Limitation of a Pathname to a Restricted Directory ('Path Traversal')
Mod	High	DiD	Ltd		CWE-78: Improper Sanitization of Special Elements used in an OS Command ('OS Command Injection')
Mod	High		Ltd		CWE-79: Failure to Preserve Web Page Structure ('Cross-site Scripting')
Mod	High	DiD	Ltd		CWE-89: Improper Sanitization of Special Elements used in an SQL Command ('SQL Injection')
Mod		DiD	Ltd		CWE-98: Improper Control of Filename for Include/Require Statement in PHP Program ('PHP File Inclusion')
Mod		DiD	Ltd		CWE-120: Buffer Copy without Checking Size of Input ('Classic Buffer Overflow')
High		DiD	Ltd		CWE-129: Improper Validation of Array Index
Mod		DiD	Ltd		CWE-131: Incorrect Calculation of Buffer Size
Mod		DiD	Ltd		CWE-190: Integer Overflow or Wraparound
Ltd	High	DiD	Mod		CWE-209: Information Exposure Through an Error Message
		DiD	Mod	Mod	CWE-285: Improper Access Control (Authorization)
		Mod		Mod	CWE-306: Missing Authentication for Critical Function
		DiD			CWE-311: Missing Encryption of Sensitive Data
				High	CWE-327: Use of a Broken or Risky Cryptographic Algorithm
			Ltd		CWE-352: Cross-Site Request Forgery (CSRF)
		DiD			CWE-362: Race Condition
Mod		DiD	Mod		CWE-434: Unrestricted Upload of File with Dangerous Type
		DiD			CWE-494: Download of Code Without Integrity Check
Mod	Mod		Ltd		CWE-601: URL Redirection to Untrusted Site ('Open Redirect')
	Ltd	DiD		Mod	CWE-732: Incorrect Permission Assignment for Critical Resource
Mod	Ltd	DiD			CWE-754: Improper Check for Unusual or Exceptional Conditions
Ltd		DiD	Ltd		CWE-770: Allocation of Resources Without Limits or Throttling
		DiD	High	Mod	CWE-798: Use of Hard-coded Credentials
Mod		DiD	Ltd		CWE-805: Buffer Access with Incorrect Length Value
Mod		DiD	Mod	Mod	CWE-807: Reliance on Untrusted Inputs in a Security Decision

Focus Profiles

The prioritization of items in the general Top 25 list is just that - general. The rankings, and even the selection of which items should be included, can vary widely depending on context. Ideally, each organization can decide how to rank weaknesses based on its own criteria, instead of relying on a single general-purpose list.

A separate document provides several "focus profiles" with their own criteria for selection and ranking, which may be more useful than the general list.

Name	Description
On the Cusp: Weaknesses that Did Not Make the 2010 Top 25	From the original nominee list of 41 submitted CWE entries, the Top 25 was selected. This "On the Cusp" profile includes the remaining 16 weaknesses that did not make it into the final Top 25.
Educational Emphasis	This profile ranks weaknesses that are important from an educational perspective within a school or university context. It focuses on the CWE entries that graduating students should know, including historically important weaknesses.
Weaknesses by Language	This profile specifies which weaknesses appear in which programming languages. Notice that most weaknesses are actually language-independent, although they may be more prevalent in one language or another.
Weaknesses Typically Fixed in Design or Implementation	This profile lists weaknesses that are typically fixed in design or implementation.
Automated vs. Manual Analysis	This profile highlights which weaknesses can be detected using automated versus manual analysis. Currently, there is very little public, authoritative information about the efficacy of these methods and their utility. There are many competing opinions, even among experts. As a result, these ratings should only be treated as guidelines, not rules.
Weaknesses by Language	This profile specifies which weaknesses appear in which programming languages. Notice that most weaknesses are actually language-independent, although they may be more prevalent in one language or another.
For Developers with Established Software Security Practices	This profile is for developers who have already established security in their practice. It uses votes from the major developers who contributed to the Top 25.
Ranked by Importance - for Software Customers	This profile ranks weaknesses based primarily on their importance, as determined from the base voting data that was used to create the general list. Prevalence is included in the scores, but it has much less weighting than importance.
Weaknesses by Technical Impact	This profile lists weaknesses based on their technical impact, i.e., what an attacker can accomplish by exploiting each weakness.

Background Details to Check Out

cwe.mitre.org/top25

- Process description
- Changelog for each revision
- On the Cusp – weaknesses that almost made it
- Appendices
  - **Selection Criteria and Supporting Fields**
  - **Threat Model for the Skilled, Determined Attacker**

Making Security Measurable™

Frequently Asked Questions (FAQ)

How is this different from the OWASP Top Ten?

The short answer is that the OWASP Top Ten covers more general concepts and is focused on web applications. The CWE Top 25 covers a broader range of issues than what arise from the web-centric view of the OWASP Top Ten, such as buffer overflows. Also, one goal of the CWE Top 25 is to be at a level that is directly actionable to programmers, so it contains more detailed issues than the categories being used in the Top Ten. There is some overlap, however, since web applications are so prevalent, and some issues in the Top Ten have general applications to all classes of software.

How are the weaknesses prioritized on the list?

With the exception of Input Validation being listed as number 1 (partially for educational purposes), there is no concrete prioritization. Prioritization differs widely depending on the audience (e.g. web application developers versus OS developers) and the risk tolerance (whether code execution, data theft, or denial of service are more important). It was also believed that the use of categories would help the organization of the document, and prioritization would impose a different ordering.

Why are you including overlapping concepts like input validation and XSS, or incorrect calculation and buffer overflows? Why do you have mixed levels of abstraction?

While it would have been ideal to have a fixed level of abstraction and no overlap between weaknesses, there are several reasons why this was not achieved.

Contributors sometimes suggested different CWE identifiers that were closely related. In some cases, this difference was addressed by using a more abstract CWE identifier that covered the relevant cases.

In other situations, there was strong advocacy for including lower-level issues such as SQL injection and cross-site scripting, so these were added. The general trend, however, was to use more abstract weakness types.

While it might be desired to minimize overlap in the Top 25, many vulnerabilities actually deal with the interaction of 2 or more weaknesses. For example, external control of user state data (CWE-642) could be an important weakness that enables cross-site scripting (CWE-79) and SQL injection (CWE-89). To eliminate overlap in the Top 25 would lose some of this important subtlety.

Finally, it was a conscious decision that if there was enough prevalence and severity, design-related weaknesses would be included. These are often thought of as being more abstract than weaknesses that arise during implementation.

The Top 25 list tries to strike a delicate balance between usability and relevance, and we believe that it does so, even with this apparent imperfection.

Why don't you use hard statistics to back up your claims?

The appropriate statistics simply aren't publicly available. The publicly available statistics are either too high-level or not comprehensive enough. And none of them are comprehensive across all software types and environments.

People are Starved for Simplicity

The Top 25 is not…

- A silver bullet
- A guarantee of software health
- A perfect match for your unique needs
- As simple as it seems
- The only thing to include in contract language
- Completely found by tools

Making Security Measurable™

The Top 25 is…

- A mechanism for awareness
- A trigger of questions
- A place for mitigations
- A conversation starter
- A first step on the long road to software assurance

Making Security Measurable™

CWE Top 25 for 2011

- Started last month
- Utilizing the Common Weakness Scoring System (CWSS 0.4) and the Common Weakness Risk Assessment Framework (CWRAF 0.4) as under-pinning
- Will have numerous “Top 10’s” & one “Top 25”
  - **Including Web, Embedded, e-Voting,…**
- Final "master" Top 25 list, will leverage combined score from multiple vignettes.
- No fixed date for release of the 2011 Top 25 at this point, may take 2 to 3 months.

Making Security Measurable™

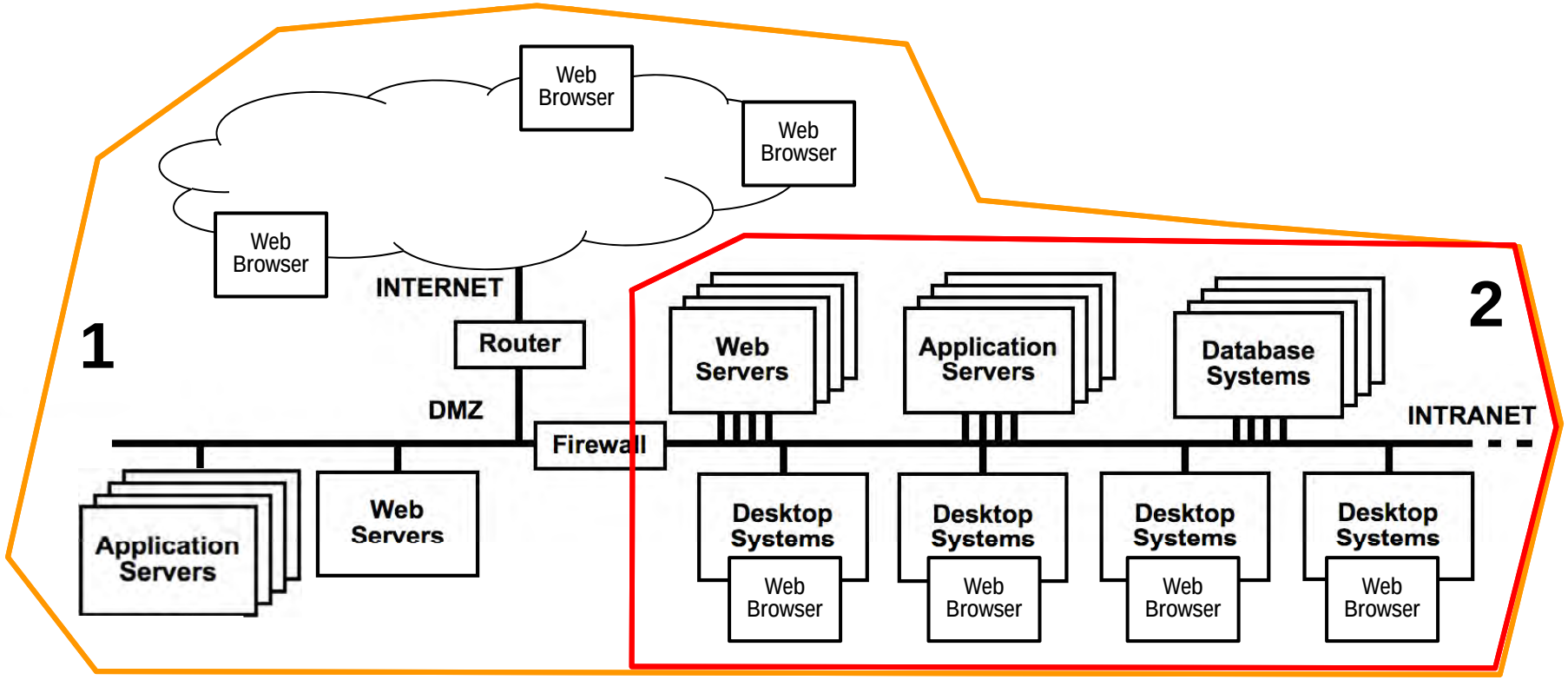
Common Weakness Scoring System (CWSS)

Archetypes:

- **Web Browser User Interface**
- **Web Servers**
- **Application Servers**
- **Database Systems**
- **Desktop Systems**
- **SSL**

Vignettes:

1. **Web-based Retail Provider**
2. **Intranet resident health records management system of hospital**



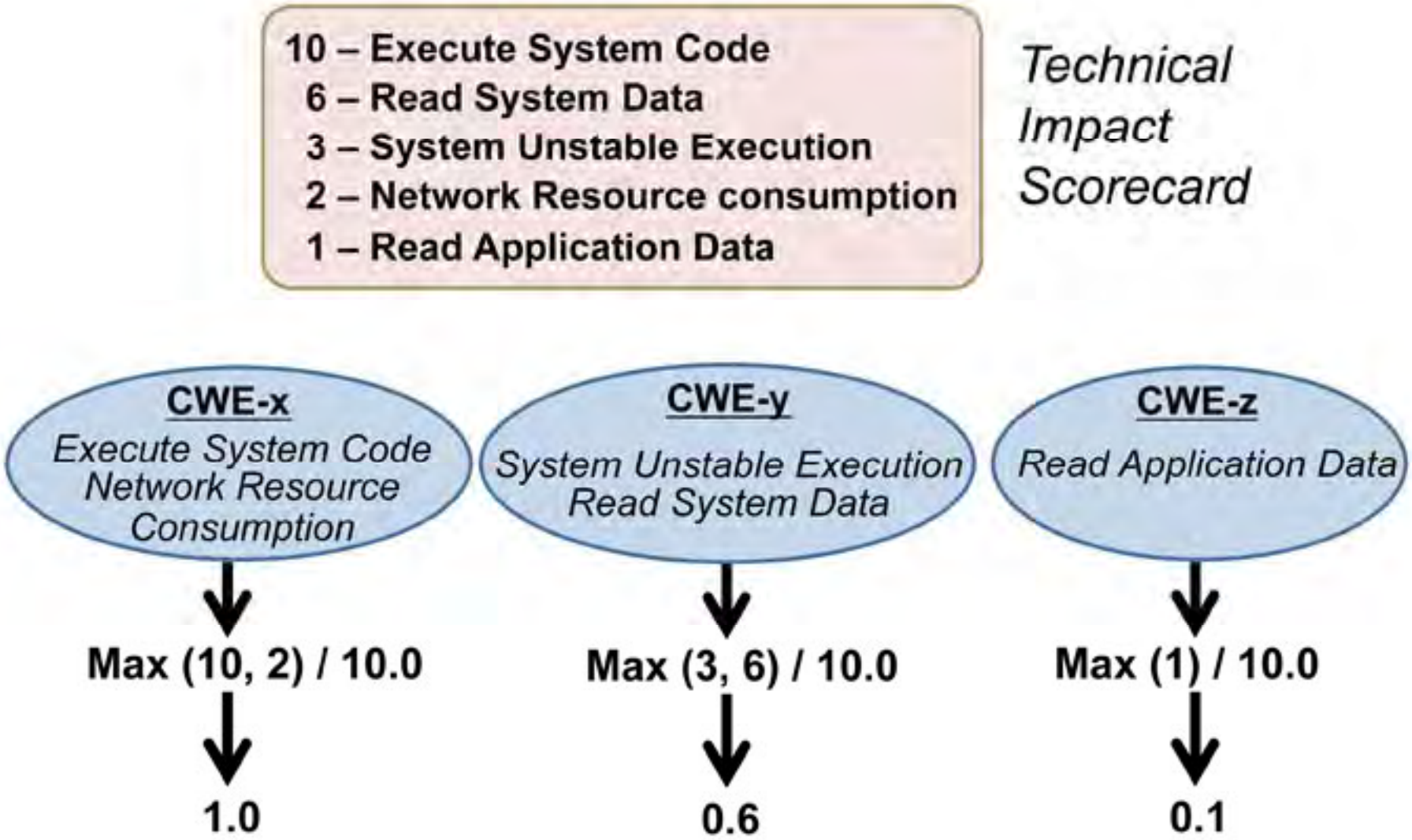
Business Value Context (BVC)

- Identifies critical assets and security concerns
- Links Technical Impacts (derived from CWE weaknesses) with business implications
- More fine-grained model than the CIA Triad

CWE Technical Impacts

1. **Modify memory**
2. **Read memory**
3. **Modify files or directories**
4. **Read files or directories**
5. **Modify application data**
6. **Read application data**
7. **DoS: crash / exit / restart**
8. **DoS: amplification**
9. **DoS: instability**
10. **DoS: resource consumption (CPU)**
11. **DoS: resource consumption (memory)**
12. **DoS: resource consumption (other)**
13. **Execute unauthorized code or commands**
14. **Gain privileges / assume identity**
15. **Bypass protection mechanism**
16. **Hide activities**

Calculating CWSS Impact Weights



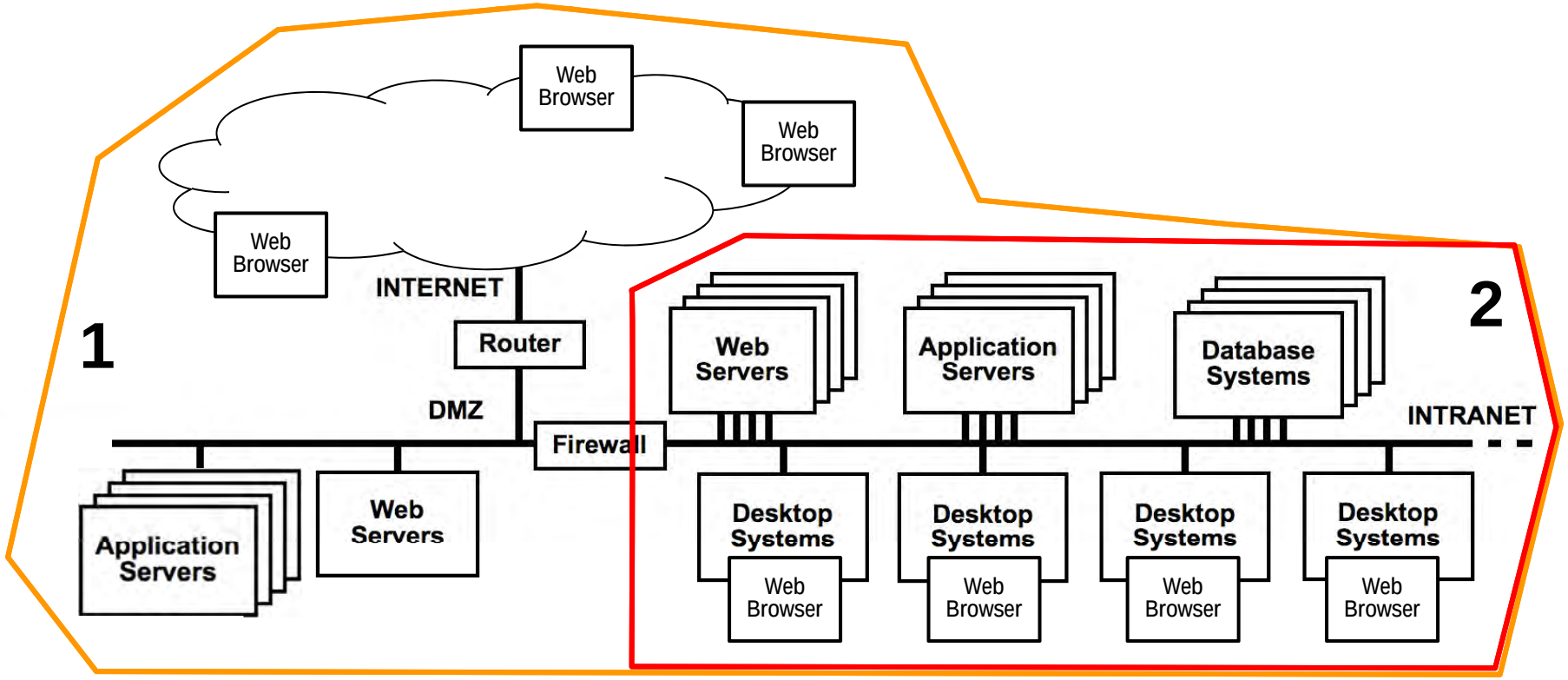
Common Weakness Scoring System (CWSS)

Archetypes:

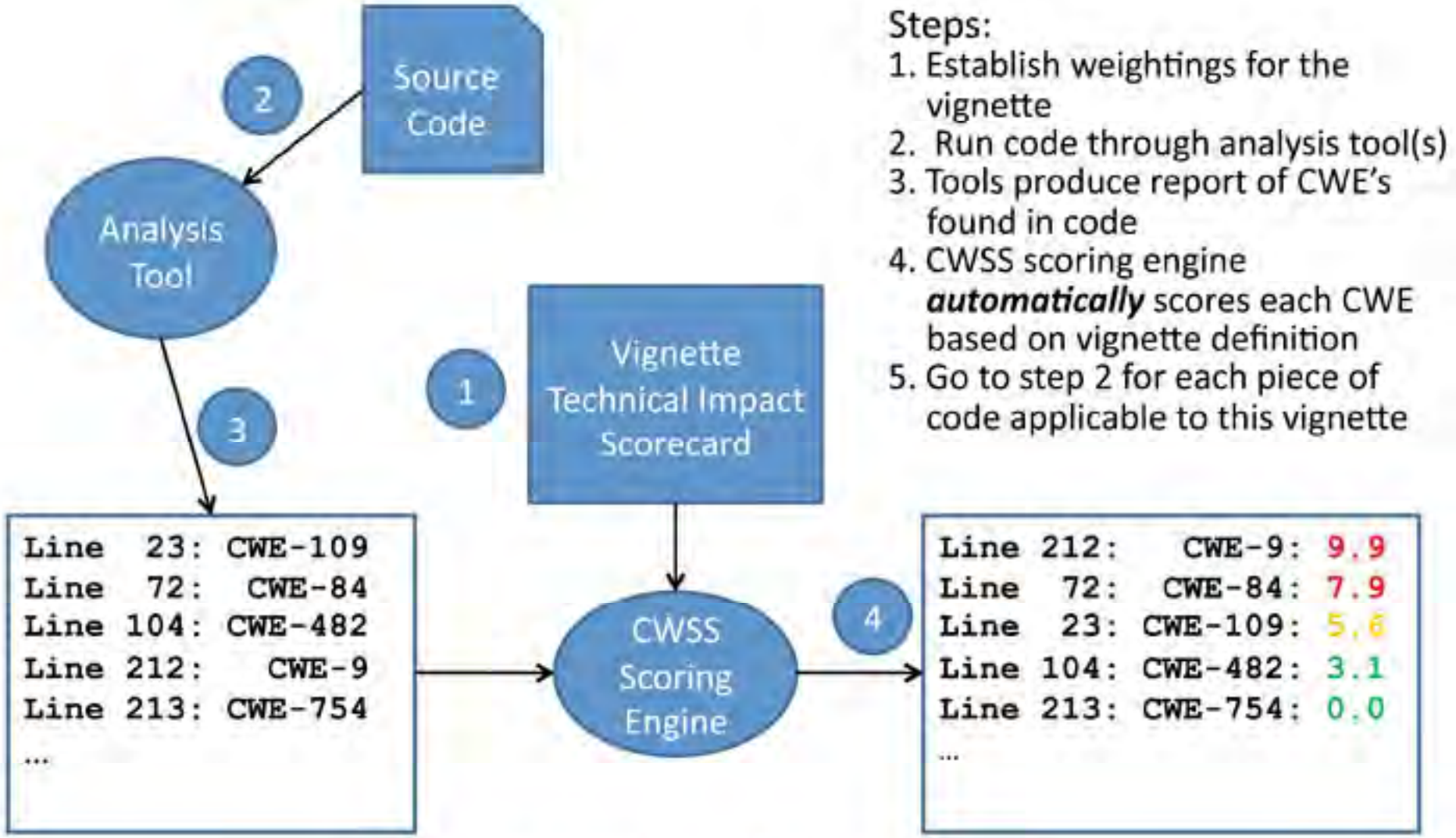
- **Web Browser User Interface**
- **Web Servers**
- **Application Servers**
- **Database Systems**
- **Desktop Systems**
- **SSL**

Vignettes:

1. **Web-based Retail Provider**
2. **Intranet resident health records management system of hospital**



Scoring Weaknesses Discovered in Code using CWSS



Steps:

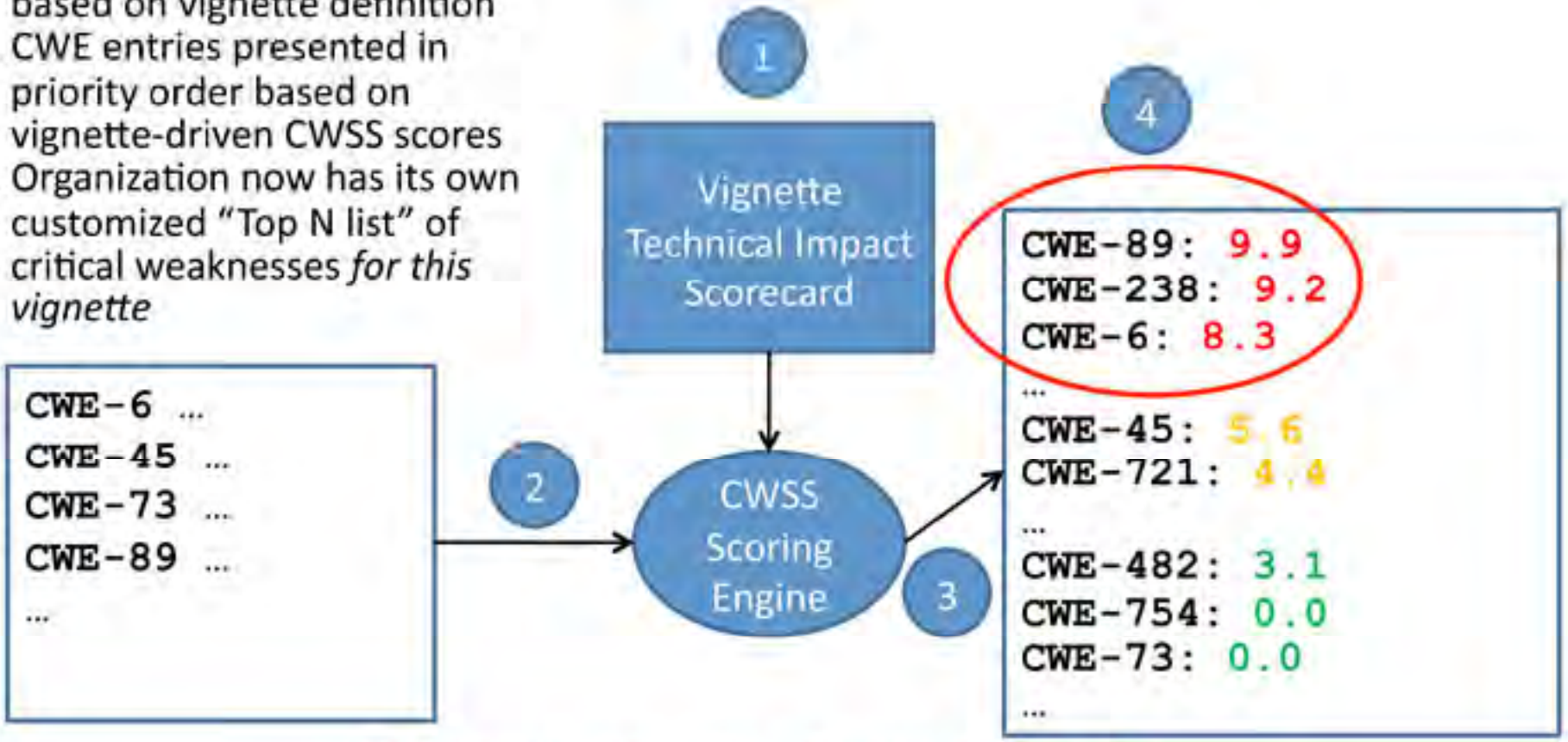
1. Establish weightings for the vignette
2. Run code through analysis tool(s)
3. Tools produce report of CWE's found in code
4. CWSS scoring engine ***automatically*** scores each CWE based on vignette definition
5. Go to step 2 for each piece of code applicable to this vignette

Step 1 is only done once – the rest is automatic

Scoring Relevant Weaknesses using CWSS

Steps:

1. Establish weightings for the vignette
2. CWSS scoring engine processes each relevant CWE entry and *automatically* scores the entry based on vignette definition
3. CWE entries presented in priority order based on vignette-driven CWSS scores
4. Organization now has its own customized "Top N list" of critical weaknesses *for this vignette*



Step 1 is only done once – the rest is automatic

CWSS for a Technology Group

50%	Web Vignette 1 … TI(1), TI(2), TI(3),…	Top N List 1
10%	Web Vignette 2 … TI(1), TI(2), TI(3),…	Top N List 2
10%	Web Vignette 3 … TI(1), TI(2), TI(3),…	Top N List 3
10%	Web Vignette 4 … TI(1), TI(2), TI(3),…	Top N List 4
15%	Web Vignette 5 … TI(1), TI(2), TI(3),…	Top N List 5
15%	Web Vignette 6 … TI(1), TI(2), TI(3),…	Top N List 6
	Web Application Technology Group	Top 10 List

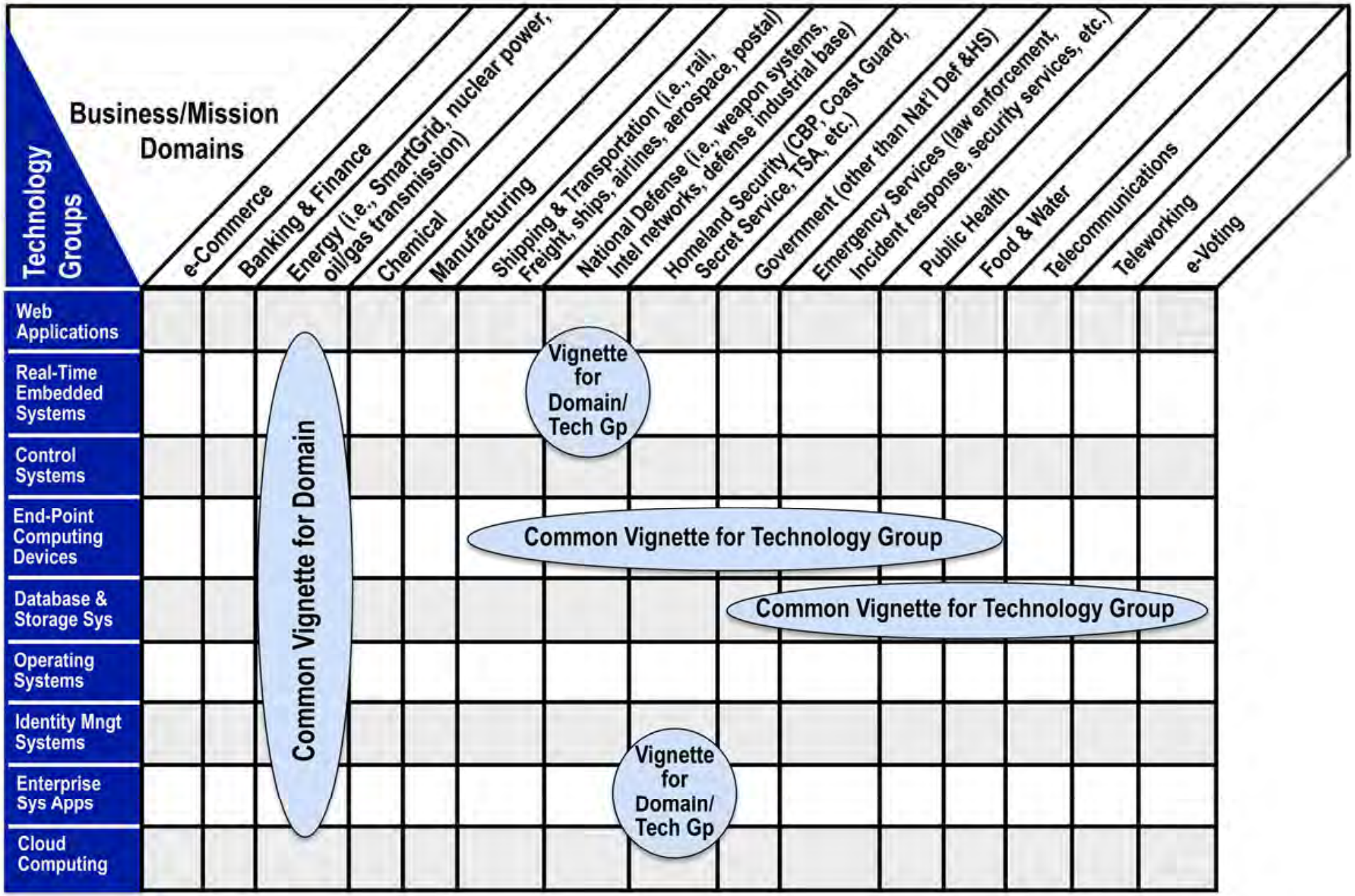
CWE Top 10 List for Web Applications can be used to:

- **Identify skill and training needs for your web team**
- **Include in T's & C's for contracting for web development**
- **Identify tool capability needs to support web assessment**

Technology Group	Archetypes/Description
Web Applications	Web browser, web-server, web-based applications and services, etc.
Industrial Control Systems	SCADA, process control system, etc.
Real-time, Embedded Systems	Embedded Device, Programmable logic controller, implanted medical devices, avionics package.
End-point Computing Devices	Smart phone, laptop, personal digital assistant (PDA), and other remote devices that leave the enterprise and/or connect remotely to the enterprise.
Cloud Computing	Hosted applications or capabilities provided over the Internet, including Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS), and Infrastructure as a Service (IaaS).
Operating Systems	General-purpose OS, virtualized OS, Real-time operating system (RTOS), hypervisor, microkernel.
Enterprise Desktop Applications/Systems	Office products such as word processing, spreadsheets, project management, etc.

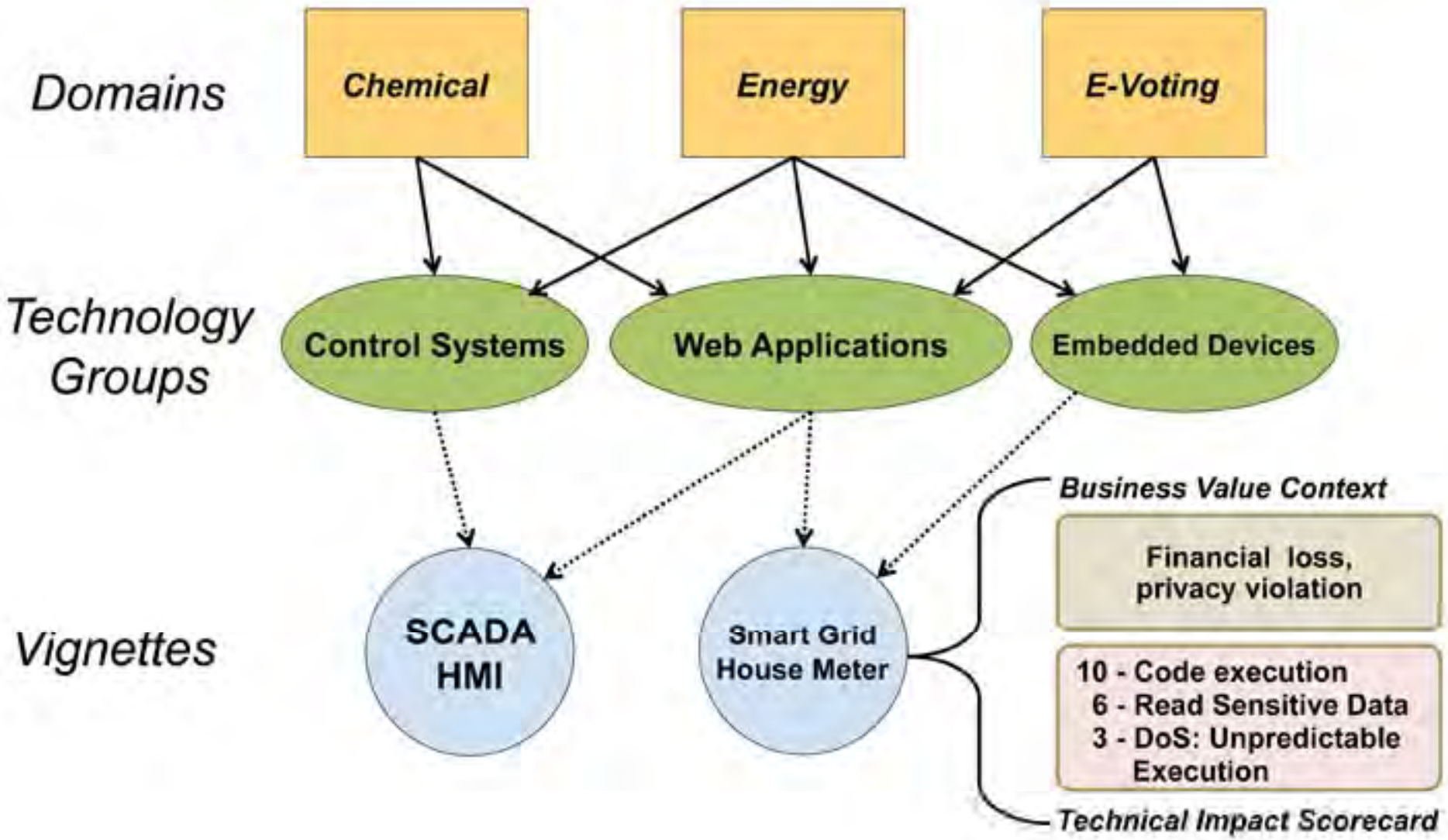
Domain Name	Description
E-Commerce	The use of the Internet or other computer networks for the sale of products and services, typically using on-line capabilities.
Banking & Finance	Financial services, including banks, stock exchanges, brokers, investment companies, financial advisors, and government regulatory agencies.
Public Health	Health care, medical encoding and billing, patient information/data, critical or emergency care, medical devices (implantable, partially embedded, patient care), drug development and distribution, food processing, clean water treatment and distribution (including dams and processing facilities), etc.
Energy	Smart Grid (electrical network through a large region, using digital technology for monitoring or control), nuclear power stations, oil and gas transmission, etc.
Chemical	Chemical processing and distribution, etc.
Manufacturing	Plants and distribution channels, supply chain, etc.
Shipping & Transportation	Aerospace systems (such as safety-critical ground aviation systems, on-board avionics, etc), shipping systems, rail systems, etc.
National Security	National security systems (including networks and weapon systems), Defense Industrial Base, etc.
Government and Commercial Security	Homeland Security systems, commercial security systems, etc.
Emergency Services	Systems and services that support first responders, incident management and response, law enforcement, and emergency services for citizens, etc.
Telecommunications	Cellular services, land lines, VOIP, cable & fiber networks, etc.
Telecommuting & Teleworking	Support for employees to have remote access to internal business networks and capabilities.
eVoting	Electronic voting systems, as used within state-run elections, shareholder meetings, etc.

Vignettes – Technology Groups & Business/Mission Domains

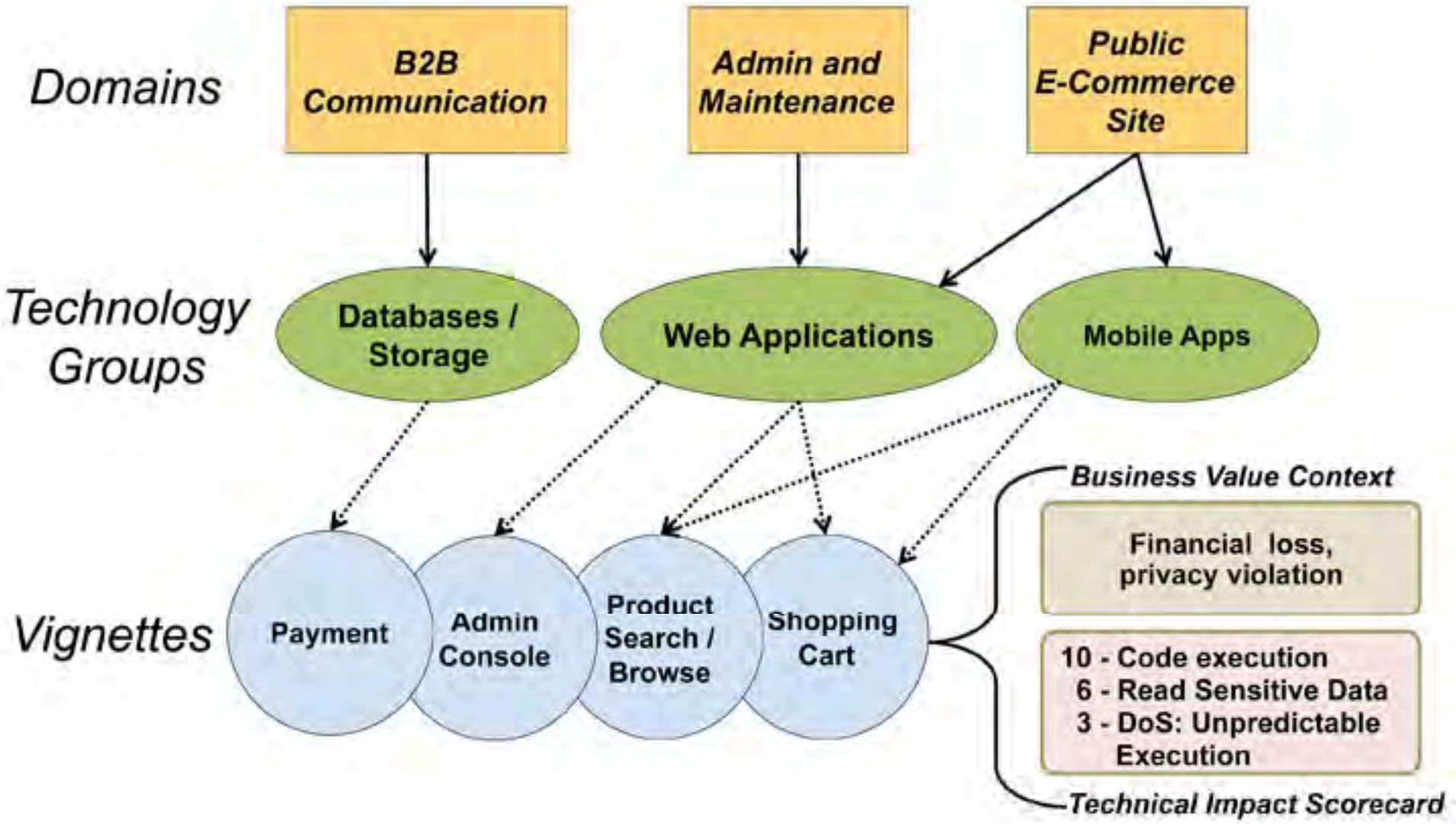


Common Weakness Risk Assessment Framework uses Vignettes with Archetypes to identify top CWEs in respective Domain/Technology Groups

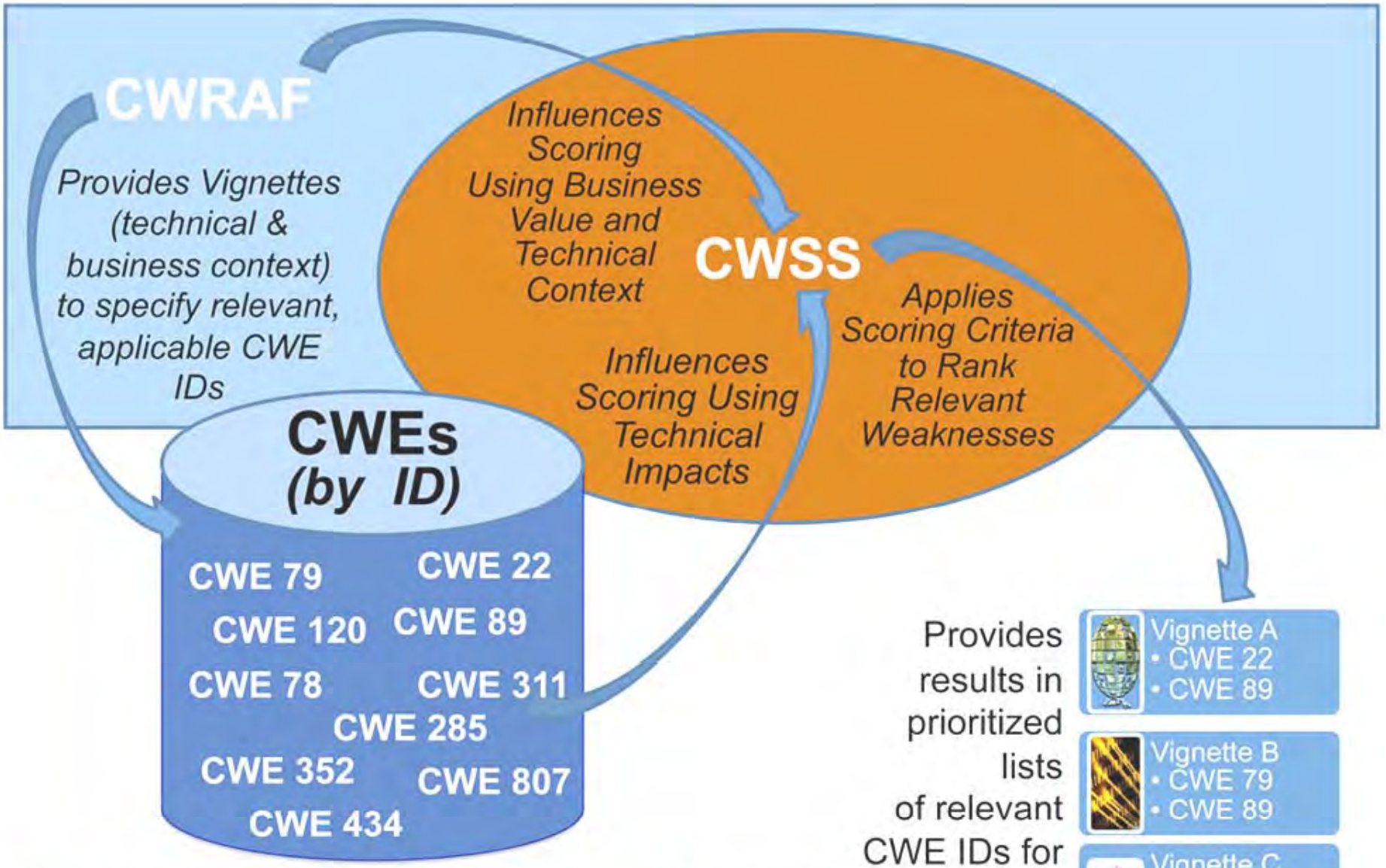
CWRAF: Common Weakness Risk Analysis Framework



Customizing CWRAF to a Single In-house Software Package



Relationships between CWRAF, CWSS, and CWE



Note: CWSS can be used in the context of CWRAF; but it is not a requirement.


Questions?
ramartin@mitre.org